```
$SAVE
$NOLIST

/*
 *      ASTLIB.H
 *
 *          This header describes the public entries in
 *          the assembly language assist library, ASTLIB.
 *
 */


D$XMTC:         PROCEDURE(c)                    EXTERNAL;
  DECLARE       c               BYTE;
  END;


FLAGS:          PROCEDURE                       WORD EXTERNAL;
  END;


$       IF EXTENDED_MONITOR
GFB:            PROCEDURE(base,offset)          BYTE EXTERNAL;
  DECLARE       base            WORD,
                offset          WORD;
  END;
$       ENDIF


INCB:           PROCEDURE(byte$ptr)             EXTERNAL;
  DECLARE       byte$ptr        POINTER;
  END;


$       IF EXTENDED_MONITOR
SFB:            PROCEDURE(val,base,offset)      EXTERNAL;
  DECLARE       val             BYTE,
                base            WORD,
                offset          WORD;
  END;
$       ENDIF


S$XMTC:         PROCEDURE(c)                    EXTERNAL;
  DECLARE       c               BYTE;
  END;


VIDEO_INTR_A:   PROCEDURE                       EXTERNAL;
  END;


XEC:            PROCEDURE(reg$p)                EXTERNAL;
  DECLARE       reg$p           POINTER;
  END;


INIT_ITV:       PROCEDURE(ivi,ivp)              EXTERNAL;
  DECLARE       ivi             BYTE,
                ivp             POINTER;
  END;

$RESTORE
```

```
$SUBTITLE('Compiler Controls')
$OPTIMIZE(3)
$NOOVERFLOW
$COMPACT
$ROM
$XREF
$LARGE( MTR100 EXPORTS  MTR_MON;
$       EXPORTS         MTR_MON;
$       EXPORTS         MTR_SWIM;
$       EXPORTS         MTR_DCRT;
$       EXPORTS         MTR_DKBD;
$       EXPORTS         MTR_SCRT;
$       EXPORTS         MTR_SKBD;
$       EXPORTS         MTR_TTY_INTR;
$       EXPORTS         MTR_TTY_POLL;
$       EXPORTS         MTR_IRET)
$LARGE( VIDEOA EXPORTS  MTR_DCI;
$       EXPORTS         MTR_DFC;
$       EXPORTS         MTR_EDC;
$       EXPORTS         MTR_FONT;
$       EXPORTS         MTR_MDC;
$       EXPORTS         MTR_MDL;
$       EXPORTS         MTR_PROMPT;
$       EXPORTS         MTR_RDC;
$       EXPORTS         MTR_UIES;
$       EXPORTS         MTR_XCA)
$LARGE( FONT   EXPORTS  MTR_FONT)
$LARGE( ASTLIB  EXPORTS VIDEO_INTR_A)
$RESET(EXTENDED_MONITOR)
```

```
DC      CR_DSAL         LT      '0DH'           /* Start  Low           */;
DC      CR_CSAH         LT      '0EH'           /* Cursor Start High    */;
DC      CR_CSAL         LT      '0FH'           /* Cursor Start Low     */;

/*
 *      Cursor Start Register Fields
 */

DC      CSR_CSD         LT      '00$11111B'     /* Cursor Start Display */;
DC      CSR_OFF         LT      '01$00000B'     /* Disable Cursor Disp  */;
DC      CSR_BLINK16     LT      '10$00000B'     /* Blink at 1/16 Field  */;
DC      CSR_BLINK       LT      '11$00000B'     /* Blink/Enable Field   */;

$RESTORE
```

```
$SAVE
$NOLIST

DECLARE HEX_DIGIT(*)    BYTE EXTERNAL DATA;

CRLF:           PROCEDURE                       EXTERNAL;
  END;

$IF 1=2
GFW:            PROCEDURE(base,offset)          WORD EXTERNAL;
  DECLARE       base            WORD,
                offset          WORD;
  END;
$ENDIF

$ IF EXTENDED_MONITOR
HCTB:           PROCEDURE(char)                 BYTE EXTERNAL;
  DECLARE       char            BYTE;
  END;
$ ENDIF

$ IF EXTENDED_MONITOR
IHC:            PROCEDURE(delim)                BYTE EXTERNAL;
  DECLARE       delim           BYTE;
  END;
$ ENDIF

$ IF EXTENDED_MONITOR
IHV:            PROCEDURE(delim,val$p,size)     BYTE EXTERNAL;
  DECLARE       delim           BYTE,
                val$p           POINTER,
                size            BYTE;
  END;
$ ENDIF

MCU:    PROCEDURE(char)                         BYTE EXTERNAL;
  DECLARE       char            BYTE;
  END;

$ IF EXTENDED_MONITOR
OHB:            PROCEDURE(dat)                  EXTERNAL;
  DECLARE       dat             BYTE;
  END;
$ ENDIF

$ IF EXTENDED_MONITOR
OHC:            PROCEDURE(dat)                  EXTERNAL;
  DECLARE       dat             BYTE;
  END;
$ ENDIF

$ IF EXTENDED_MONITOR
OHW:            PROCEDURE(dat)                  EXTERNAL;
  DECLARE       dat             WORD;
```

```
$SAVE
$NOLIST

/*
 *      GLOBAL.H
 *
 *          GLOBAL.DEF is an include file to define all of
 *          the global values in the data segment.  This
 *          file should be included in only one source file
 *          to avoid multiple definitions.  To define all
 *          of these variables as external to some other
 *          module, include GLOBAL.H.
 *
 */


/*
 *      Monitor Parameters
 */

DC      CODE_SEG        LIT     '0FE05H';       /* Code Segment         */
DC      MTR_VERSION     LIT     '12H';          /* Version 1.2          */
DC      MTR_ISSUE       LIT     '54H';          /* Issue   #nn          */


DC      WIP(5)  BYTE            /* Jump Far to Wild Interrupt Routine   */
                EXTERNAL;
DC      VERSION BYTE            /* BCD Version Identification for ROM   */
                EXTERNAL;
DC      DS_SIZE WORD            /* Data Segment Size in bytes           */
                EXTERNAL;


/*
 *      Boot Parameters
 */

DC      BOOT_PAR        STRUCTURE (
            INDEX               BYTE,   /* Device Index                 */
            PORT                BYTE,   /* Base Port Address            */
            STRNG(80)           BYTE,   /* Parameter String Passed      */
            UNIT                BYTE    /* Unit of device booted        */
            )   EXTERNAL;


/*
 *      RAM Vectors for parametrized routines.
 */

DC      DCI     POINTER         /* Display Character Initialization     */
                EXTERNAL;
DC      DFC     POINTER         /* Display Font Character               */
                EXTERNAL;
DC      D_XMTC  POINTER         /* Dumb Keyboard Transmit Character     */
                EXTERNAL;
DC      EDC     POINTER         /* Erase Display Character              */
                EXTERNAL;
DC      EMEC    POINTER         /* Extended-Mode Escape Character       */
                EXTERNAL;
DC      FONT    POINTER         /* Character FONT Address               */
                EXTERNAL;
DC      MDC     POINTER         /* Move Display Characters              */
                EXTERNAL;
```

```
        )   EXTERNAL;
DC      CRTC_DISPLAY    STRUCTURE (
        START                   WORD,   /* CRT-C Display Start Address  */
        UPDATE                  BOOL    /* True if Update is required   */
        )   EXTERNAL;
DC      ESCP            STRUCTURE (
        COMMAND                 BYTE,   /* Current Escape Command       */
        FUNCTION                WORD,   /* Pointer to Escape Processor  */
        MODE                    BYTE,   /* Escape Emulation Mode        */
        OPER_COUNT              BYTE,   /* Operand Count                */
        OPER_INDEX              BYTE,   /* Operand Index                */
        OPERAND(4)              BYTE    /* Operands                     */
        )   EXTERNAL;
DC      H19_MODE        STRUCTURE (
        ALTERNATE               BOOL,   /* Alternate Key-Pad Mode       */
        ANSI                    BOOL,   /* ANSI-Mode                    */
        AUTO_CR                 BOOL,   /* Auto Carriage-Return on LF   */
        AUTO_LF                 BOOL,   /* Auto Line-Feed on CR         */
        AUTO_REPEAT             BOOL,   /* Auto-Repeat Keyboard         */
        BWO                     BOOL,   /* Black and White Optimization */
        CURSOR                  BYTE,   /* Cursor Program Value         */
        CURSOR_ON               BOOL,   /* Cursor Enabled               */
        EXPAND                  BOOL,   /* Expand KeyBoard Characters   */
        GRAPHIC                 BOOL,   /* Graphic Character Mode       */
        INSERT                  BOOL,   /* Insert Character Mode        */
        KEY_EN                  BOOL,   /* Key-Board Enabled            */
        REVERSE                 WORD,   /* Reverse Video                */
                                        /*   00000H for normal video    */
                                        /*   0FFFFH for reverse video   */
        SHIFTED                 BOOL,   /* Shifted Key-Pad Mode         */
        STATUS                  BOOL,   /* Status Line Enabled          */
        UPDN                    BOOL,   /* Key-Board Up/Down Mode       */
        WRAP                    BOOL    /* Wrap Line at End-of-Line     */
        ) EXTERNAL;
DC      H19_PAR         STRUCTURE (
        CPL                     BYTE,   /* Characters Per Line          */
        DSC                     BYTE,   /* Displayed Scan Lines/Char    */
        LPS                     BYTE,   /* Lines Per Screen             */
        PDVRAM                  BYTE,   /* Planes of Video RAM          */
        SLI                     BYTE,   /* Status Line Index            */
        SPC                     BYTE,   /* Scan Line per Character      */
        SW401                   BYTE,   /* H-19 Switch 401              */
        SW402                   BYTE,   /* H-19 Switch 402              */
        VRAM_SIZE               BYTE    /* 0-32KBytes, 1-64KBytes       */
        ) EXTERNAL;
DC      XMT             STRUCTURE (
        BURST                   BYTE,   /* Char. to XMT per VSYNC       */
        BCOUNT                  INTEGER,/* Remaining Char. in Burst     */
        COUNT                   WORD,   /* Characters to Transmit       */
        COL                     BYTE,   /* Horizontal Column to XMT     */
        ROW                     BYTE,   /* Vertical Row to XMT          */
        COLOR                   BYTE,   /* State of COLOR   XMT'd       */
        GRAPHIC                 BOOL,   /* State of GRAPHIC XMT'd       */
        REVERSE                 BOOL    /* State of REVERSE XMT'd       */
        ) EXTERNAL;

$RESTORE
```

```
$SAVE
$NOLIST

/*
 *      I8086.H
 *
 *          I8086.H defines hardware attributes of the
 *          Intel 8086 micro-processor.
 *
 */

DC      F86_OF          LT      '1000$0000$0000B';      /* Overflow     */
DC      F86_DF          LT      '0100$0000$0000B';      /* Direction    */
DC      F86_IF          LT      '0010$0000$0000B';      /* Interrupt En */
DC      F86_TF          LT      '0001$0000$0000B';      /* Trap         */
DC      F86_SF          LT      '0000$1000$0000B';      /* Sign         */
DC      F86_ZF          LT      '0000$0100$0000B';      /* Zero         */
DC      F86_AF          LT      '0000$0001$0000B';      /* Aux. Carry   */
DC      F86_PF          LT      '0000$0000$0100B';      /* Parity       */
DC      F86_CF          LT      '0000$0000$0001B';      /* Carry        */

$RESTORE
```

```
$SAVE
$NOLIST

/*
 *       INIT.H
 *
 *           INIT.H defines the gloabal initialization
 *           routines in INIT.PLM
 *
 */


$ IF EXTENDED_MONITOR
INIT_8259_85:           PROCEDURE       EXTERNAL;
  END;
$ ENDIF


INIT_8259_88:           PROCEDURE       EXTERNAL;
  END;

$RESTORE
```

```
$SAVE
$NOLIST

$       IF EXTENDED_MONITOR

I2661:  PROCEDURE(base)         EXTERNAL;
  DECLARE       base            WORD;
END;


RCHAR:  PROCEDURE(base)         BYTE EXTERNAL;
  DECLARE       base            WORD;
END;


TCHAR:  PROCEDURE(base)         BOOL EXTERNAL;
  DECLARE       base            WORD;
  END;


WCHAR:  PROCEDURE(base,char)    EXTERNAL;
  DECLARE       base            WORD,
                char            BYTE;
END;

$       ENDIF

$RESTORE
```

```
DC      ICW4_AEOI       LT      '0000$0010B'    /* Auto. End Of Int.    */;
DC      ICW4_86         LT      '0000$0001B'    /* MCS-86 Operation     */;

/*
 *      Output Control Words
 */

DC      OCW1_M7         LT      '1000$0000B'    /* Mask Level 7         */;
DC      OCW1_M6         LT      '0100$0000B'    /* Mask Level 6         */;
DC      OCW1_M5         LT      '0010$0000B'    /* Mask Level 5         */;
DC      OCW1_M4         LT      '0001$0000B'    /* Mask Level 4         */;
DC      OCW1_M3         LT      '0000$1000B'    /* Mask Level 3         */;
DC      OCW1_M2         LT      '0000$0100B'    /* Mask Level 2         */;
DC      OCW1_M1         LT      '0000$0010B'    /* Mask Level 1         */;
DC      OCW1_M0         LT      '0000$0001B'    /* Mask Level 0         */;
DC      OCW1_ALL        LT      '1111$1111B'    /* Mask ALL Levels      */;

DC      OCW2_ROT        LT      '1000$0000B'    /* Rotate               */;
DC      OCW2_SEOI       LT      '0100$0000B'    /* Specific EOI         */;
DC      OCW2_EOI        LT      '0010$0000B'    /* End Of Interrupt     */;
DC      OCW2_OCW2       LT      '0000$0000B'    /* OCW2 Identifier      */;
DC      OCW2_L2         LT      '0000$0100B'    /* SEOI Interrupt Level */;
DC      OCW2_L1         LT      '0000$0010B'    /* SEOI Interrupt Level */;
DC      OCW2_L0         LT      '0000$0001B'    /* SEOI Interrupt Level */;

DC      OCW3_ESMM       LT      '0100$0000B'    /* Enable Special Mask  */;
DC      OCW3_SMM        LT      '0010$0000B'    /* Special Mask Mode    */;
DC      OCW3_OCW3       LT      '0000$1000B'    /* OCW3 Identifier      */;
DC      OCW3_POLL       LT      '0000$0100B'    /* Enable Polling       */;
DC      OCW3_SRIS       LT      '0000$0010B'    /* */;

$RESTORE
```

```
$SAVE
$NOLIST

/*
 *      KEYDEF.H
 *
 *          This header file defines the bits and offsets per-
 *          tinent to the key-board processor.
 *
 */


/*
 *      Port Definitions
 */

DC      KEY_CMND        LT      'IO_KEYBRD+1'   /* Command              */;
DC      KEY_DATA        LT      'IO_KEYBRD+0'   /* Data                 */;
DC      KEY_STAT        LT      'IO_KEYBRD+1'   /* Status               */;


/*
 *      Status Bit Definitions
 */

DC      KS_CXRDY        LT      '001B'          /* Character is ready   */;
DC      KS_RXRDY        LT      '010B'          /* Receiver Ready       */;


/*
 *      Command Definitions
 */

DC      KC_ARF          LT      '02H'           /* Auto-Repeat OFF      */;
DC      KC_ARO          LT      '01H'           /* Auto-Repeat ON       */;
DC      KC_BEP          LT      '07H'           /* Beep                 */;
DC      KC_DI           LT      '0DH'           /* Disable Interrupts   */;
DC      KC_EI           LT      '0CH'           /* Enbale  Interrupts   */;
DC      KC_KCF          LT      '04H'           /* Key Click   OFF      */;
DC      KC_KCO          LT      '03H'           /* Key Click   ON       */;
DC      KC_CFI          LT      '05H'           /* Clear FIFO           */;
DC      KC_CLK          LT      '06H'           /* Click                */;
DC      KC_KBD          LT      '09H'           /* Key-Board   DISABLE  */;
DC      KC_KBE          LT      '08H'           /* Key-Board   ENABLE   */;
DC      KC_MNS          LT      '0BH'           /* Normal Scan Mode     */;
DC      KC_MUD          LT      '0AH'           /* Key Up/Down Mode     */;
DC      KC_RES          LT      '00H'           /* Reset                */;


/*
 *      Key Code Definitions
 */

DC      KY_ENTER        LT      '0DH'           /* Enter                */;
DC      KY_HELP         LT      '095H'          /* HELP                 */;
DC      KY_F0           LT      '096H'          /* F0                   */;
DC      KY_F12          LT      '0A2H'          /* F12                  */;
DC      KY_ID_CHAR      LT      '0A3H'          /* Insert/Delete Char.  */;
DC      KY_ID_LINE      LT      '0A4H'          /* Insert/Delete Line   */;
DC      KY_UP           LT      '0A5H'          /* Arrow Up             */;
DC      KY_DOWN         LT      '0A6H'          /* Arrow Down           */;
DC      KY_RIGHT        LT      '0A7H'          /* Arrow Right          */;
DC      KY_LEFT         LT      '0A8H'          /* Arrow Left           */;
DC      KY_HOME         LT      '0A9H'          /* Home                 */;
```

```
$SAVE
$NOLIST

/*
 *      LEXCAL.H
 *
 *          LEXCAL.H defines a number of standard lexical
 *          headers.
 *
 */

DECLARE         DC      LITERALLY       'DECLARE';
DECLARE         LT      LITERALLY       'LITERALLY';

DC      BOOL            LT      'BYTE'  /* Boolean Variables            */;

DC      FALSE           LT      '000H'  /* Boolean False                */;
DC      TRUE            LT      '0FFH'  /* Boolean True                 */;

$RESTORE
```

```
$SAVE
$NOLIST

/*
 *      MTR100.H
 *
 *          MTR100.H is the file defining the globals to
 *          be found in MTR100.A86.
 *
 */

MTR_SCRT:       PROCEDURE               EXTERNAL;
  END;

MTR_SKBD:       PROCEDURE               EXTERNAL;
  END;

MTR_TTY_INTR:   PROCEDURE               EXTERNAL;
  END;

MTR_TTY_POLL:   PROCEDURE               BOOL EXTERNAL;
  END;

MTR_IRET:       PROCEDURE               EXTERNAL;
  END;

$ IF EXTENDED_MONITOR
R8085:          PROCEDURE               EXTERNAL;
  END;
$ ENDIF

SDS:            PROCEDURE               EXTERNAL;
  END;

$RESTORE
```

```
$SAVE
$NOLIST

/*
 *      SSWDEF.H
 *
 *          SSWDEF.H defines the H-19 compatible software
 *          switch settings.
 *
 */

DC
SW_401_BLOCK_CURSOR     LT      '001H'  /* Block Cursor                    */,
SW_401_KEY_CLICK        LT      '002H'  /* Key Click                       */,
SW_401_WRAP_LINE        LT      '004H'  /* Wrap Line at Right-Hand Margin*/,
SW_401_AUTO_LF          LT      '008H'  /* Auto Line-Feed on Carriage-Return*/,
SW_401_AUTO_CR          LT      '010H'  /* Auto Carriage-Return on Line-Feed*/,
SW_401_ANSI             LT      '020H'  /* ANSI Escape Processing Mode  */,
SW_401_SHIFTED          LT      '040H'  /* Key-Pad shifted Mode         */,
SW_401_50HZ             LT      '080H'  /* 50-Hertz CRT Refresh         */;

DC
SW_402_BAUD_RATE        LT      '00FH'  /* Baud Rate                       */,
SW_402_PARITY_ENABLE    LT      '010H'  /* Parity Enable                   */,
SW_402_ODD_PARITY       LT      '020H'  /* Odd Parity                      */,
SW_402_STICK_PARITY     LT      '040H'  /* Stick Parity                    */,
SW_402_FULL_DUPLEX      LT      '080H'  /* Full Duplex                     */;


$RESTORE
```

```
$SAVE
$NOLIST

/*
 *      SUBLIB.H
 *
 *          This file describes the global Subroutine Library
 *          entry points.
 *
 */

$ IF EXTENDED_MONITOR
IHA:            PROCEDURE(delim,addr$p)                 BYTE EXTERNAL;
  DECLARE       delim           BYTE,
                addr$p          POINTER;
  END;
$ ENDIF


$ IF 1=2
INA:            PROCEDURE(base$p,offset$p)              EXTERNAL;
  DECLARE       base$p          POINTER,
                offset$p        POINTER;
  END;
$ ENDIF


$ IF EXTENDED_MONITOR
IRI:            PROCEDURE                       BYTE EXTERNAL;
  END;
$ ENDIF


$ IF EXTENDED_MONITOR
OHA:            PROCEDURE(addr$p)                       EXTERNAL;
  DECLARE       addr$p          POINTER;
  END;
$ ENDIF


$ IF EXTENDED_MONITOR
RNC:            PROCEDURE(delim)                BYTE EXTERNAL;
  DECLARE       delim           BYTE;
  END;
$ ENDIF


$ IF EXTENDED_MONITOR
RUBOUT:         PROCEDURE                       EXTERNAL;
  END;
$ ENDIF


$RESTORE
```

```
XMTSC:          PROCEDURE(row,col,count)        EXTERNAL;
  DECLARE       row                 BYTE,
                col                 BYTE,
                count               WORD;
  END;

$RESTORE
```

$RESTORE

```
;;      E2661   - EPCI 2661
;
;       The following definitions are for the Signetics
;       2661 EPCI (Enhanced Programmable Communications
;       Interface).
;

SR      RECORD  DSR:1,DCD:1,FE_SYN:1,OVRN:1,PE_DLE:1,TxEMT_DSCHG:1,RxRDY:1,TxRDY:1

EP_RHR  EQU     0               ; Receiver Holding Register
EP_THR  EQU     0               ; Transmitter Holding Register

EP_STA  EQU     1               ; Status Register
EP_SYN1 EQU     1               ; SYN1
EP_SYN2 EQU     1               ; SYN2
EP_DLE  EQU     1               ; DLE

EP_MR1  EQU     2               ; Mode Register 1
EP_MR2  EQU     2               ; Mode Register 2

EP_CR   EQU     3               ; Command Register

CONSOL  EQU     0E8H            ; Console 2661
```

```
                EXTRN   CRLF:NEAR       ; PL/M-86 CR-LF
                EXTRN   INIT:NEAR       ; PL/M-86 Initialize Hardware
                EXTRN   WRITEC:NEAR     ; PL/M-86 Write Character
  CODE          ENDS
  )FI


%IF(%NES(DATA,%SEGMENT_LABEL)) THEN (
  DATA          SEGMENT WORD PUBLIC 'DATA'
                EXTRN   HORZ_CHAR:BYTE  ; Column Index
                EXTRN   VERT_LINE:BYTE  ; Row    Index
                EXTRN   REGP:DWORD      ; Pointer to Saved Processor Registers
                EXTRN   RESETF:BYTE     ; Hardware Reset Flag

                EXTRN   DCI:DWORD       ; Display Character Initialization
                EXTRN   DFC:DWORD       ; Display Font Character
                EXTRN   D_XMTC:DWORD    ; Dumb Terminal Transmit Character
                EXTRN   EDC:DWORD       ; Erase Display Character
                EXTRN   EMEC:DWORD      ; Extend-Mode Escape Character
                EXTRN   FONT:DWORD      ; Pointer to Font Table
                EXTRN   MDC:DWORD       ; Move Display Character
                EXTRN   MDL:DWORD       ; Move Display Line
                EXTRN   PROMPT:DWORD    ; Display Monitor Prompt
                EXTRN   RDC:DWORD       ; Read Display Character
                EXTRN   S_XMTC:DWORD    ; Smart Terminal Transmit Character
                EXTRN   UIES:DWORD      ; Un-Implemented Escape Sequence
                EXTRN   XCA:DWORD       ; Transmit Character Attributes

                EXTRN   BOOT_PAR:BOOT_PAR_STRUC
                EXTRN   COLOR:COLOR_STRUC
                EXTRN   CRTC_CURSOR:CRTC_STRUC
                EXTRN   CRTC_DISPLAY:CRTC_STRUC
                EXTRN   ESCP:ESCP_STRUC
                EXTRN   H19_MODE:H19_MODE_STRUC
                EXTRN   H19_PAR:H19_PAR_STRUC
                EXTRN   XMT:XMT_STRUC
  DATA          ENDS
  )FI


%IF(%NES(VIDEOA,%SEGMENT_LABEL)) THEN (
  VIDEOA        SEGMENT BYTE PUBLIC 'CODE'
  VIDEOA        ENDS
  )FI


CGROUP  GROUP   MTR100,CODE,ASTLIB,VIDEOA

$RESTORE
```

```
$EJECT
;;      Interrupt Usage
;
;       This common deck defines the MTR-100 Interrupt
;       usage.  Note that the interrupt vector for int-
;       errupt 255 is used to store a global data seg-
;       ment value.  This is somewhat undesirable to
;       reserve such a location, however, since all
;       other values are indexed off of it, it seems
;       alright.  The offset must be zero so that any
;       spurious interrupts will jump to the first addr-
;       ess of the data segment.  If these interrupts
;       are to be handled, the data segment vector may
;       be patched.
;

INTVEC  SEGMENT PARA AT PAR_INT

ITV_DBZ DD      1 DUP(?)        ; Divide by zero interrupt
ITV_SST DD      1 DUP(?)        ; Single Step
ITV_NMI DD      1 DUP(?)        ; Non-Maskable Interrupt
ITV_OBI DD      1 DUP(?)        ; One Byte Instruction
ITV_IOV DD      1 DUP(?)        ; Interrupt On Overflow

ITV_5   DD      1 DUP(?)        ; Interrupt 5
        DD      (32-6) DUP(?)   ; Reserved Interrupts
ITV_32  DD      1 DUP(?)        ; Interrupt 32
        DD      (255-33) DUP(?) ; Allocate space for available user interrupts
ITV_255 DD      1 DUP(?)        ; Interrupt 255 is used for MTR-100 DS

MTR_DS_O    EQU     WORD PTR ITV_255+0   ; MTR-100 Data Segment Offset
                                         ;  (must be 0)
MTR_DS_S    EQU     WORD PTR ITV_255+2   ; MTR-100 Data Segment

INTVEC  ENDS
```

```
$SAVE
$NOGEN

;;      Global Structure Definitions
;

BOOT_PAR_STRUC  STRUC
    INDEX       DB      ?       ; Device Index
    PORT        DB      ?       ; Base Port Address
    STRNG       DB      80 DUP(?)       ; Parameter String Passed
    UNIT        DB      ?       ; Unit of Device Booted
  BOOT_PAR_STRUC        ENDS

COLOR_STRUC     STRUC
  FORE          DB      ?       ; Foreground Color
  BACK          DB      ?       ; Background Color
  MSK           DB      ?       ; SHL(Back,3) OR Fore
  CLEAR         DB      ?       ; Color to Clear
  PAINTED       DB      ?       ; Color to Fully Paint
  FFONT         DB      ?       ; Color to set to Font Pattern
  COMP_FONT     DB      ?       ; Color to set to Complement of Font Pattern
    COLOR_STRUC ENDS

CRTC_STRUC      STRUC
    START       DW      ?       ; CRT-C ..... Start Address
    UPDATE      DB      ?       ; True if update is required
  CRTC_STRUC    ENDS

ESCP_STRUC      STRUC
    COMMAND     DB      ?       ; Current Escape Command
    FUNCTION    DW      ?       ; Pointer to Escape Processor
    MODE        DB      ?       ; Escape Emulation Mode
    OPER_COUNT  DB      ?       ; Operand Count
    OPER_INDEX  DB      ?       ; Operand Index
    OPERAND     DB      4 DUP(?); Operands
  ESCP_STRUC    ENDS

H19_MODE_STRUC  STRUC
    ALTERNATE   DB      ?       ; Alternate Key-pad Mode
    ANSI        DB      ?       ; ANSI Mode
    AUTO_CR     DB      ?       ; Auto Carriage-Return on Line-Feed
    AUTO_LF     DB      ?       ; Auto Line-Feed on Carriage-Return
    AUTO_REPEAT DB      ?       ; Auto-Repeat Keyboard
    BWO         DB      ?       ; Black and White Optimization
    CURSOR      DB      ?       ; Programmed Cursor Value
    CURSOR_ON   DB      ?       ; Cursor Enabled
    EXPAND      DB      ?       ; Expand Key-Board Characters
    GRAPHIC     DB      ?       ; Graphic Character Mode
    INSERT      DB      ?       ; Insert Character Mode
    KEY_EN      DB      ?       ; Key-Board Enable
    REVERSE     DW      ?       ; Reverse Video
    SHIFTED     DB      ?       ; Shifted Key-Pad Mode
    STATUS      DB      ?       ; Status-Line Enabled
    UPDN        DB      ?       ; Key-Board Up/Down Mode
    WRAP        DB      ?       ; Wrap at End-of-Line
  H19_MODE_STRUC        ENDS

H19_PAR_STRUC   STRUC
    CPL         DB      ?       ; Characters Per Line
    DSC         DB      ?       ; Displayed Scan Lines per Character
    LPS         DB      ?       ; Lines Per Screen
    POVRAM      DB      ?       ; Planes of Video RAM
    SLI         DB      ?       ; Status Line Index
    SPC         DB      ?       ; Scan Lines per Character
```

```
SERIES-III 8086/8087/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE GLOBAL
OBJECT MODULE PLACED IN :F2:GLOBAL.OBJ
INVOCATION LINE CONTROLS:  PRINT(:TO:) XREF ERRORPRINT(:TO:)


LOC  OBJ                  LINE     SOURCE
```

```
LOC  OBJ                  LINE     SOURCE

                            14     %*DEFINE(SEGMENT_LABEL)(DATA)
                            15     $       INCLUDE (:F2:EXTERN.COM)
                      =1    16     $SAVE
                      =1    17     $NOGEN
                      =1    18
                      =1    19     %*DEFINE(EXTENDED_MONITOR)(0)    %' EXTENDED_MODE is FALSE
                      =1
  00FF                =1    20     PLM_TRUE        EQU     0FFH    ; PL/M-86 Boolean TRUE
                      =1    21
                      =1    22     ;       Structure Definitions
                      =1    23
                      =1    24     $       INCLUDE (':F2:STRDEF.COM')
                      =2    25     $SAVE
                      =2    26     $NOGEN
                      =2    27
                      =2    28     ;;      Global Structure Definitions
                      =2    29     ;
                      =2    30
  ----                =2    31     BOOT_PAR_STRUC  STRUC
  0000                =2    32         INDEX       DB      ?       ; Device Index
  0001                =2    33         PORT        DB      ?       ; Base Port Address
  0002                =2    34         STRNG       DB      80 DUP(?)       ; Parameter String Passed
  0052                =2    35         UNIT        DB      ?       ; Unit of Device Booted
  ----                =2    36       BOOT_PAR_STRUC        ENDS
                      =2    37
  ----                =2    38     COLOR_STRUC     STRUC
  0000                =2    39       FORE          DB      ?       ; Foreground Color
  0001                =2    40       BACK          DB      ?       ; Background Color
  0002                =2    41       MSK           DB      ?       ; SHL(Back,3) OR Fore
  0003                =2    42       CLEAR         DB      ?       ; Color to Clear
  0004                =2    43       PAINTED       DB      ?       ; Color to Fully Paint
  0005                =2    44       PFONT         DB      ?       ; Color to set to Font Pattern
  0006                =2    45       COMP_FONT     DB      ?       ; Color to set to Complement of Font Pattern
  ----                =2    46         COLOR_STRUC ENDS
                      =2    47
  ----                =2    48     CRTC_STRUC      STRUC
  0000                =2    49         START       DW      ?       ; CRT-C ..... Start Address
  0002                =2    50         UPDATE      DB      ?       ; True if update is required
  ----                =2    51       CRTC_STRUC    ENDS
                      =2    52
  ----                =2    53     ESCP_STRUC      STRUC
  0000                =2    54         COMMAND     DB      ?       ; Current Escape Command
  0001                =2    55         FUNCTION    DW      ?       ; Pointer to Escape Processor
  0003                =2    56         MODE        DB      ?       ; Escape Emulation Mode
  0004                =2    57         OPER_COUNT  DB      ?       ; Operand Count
  0005                =2    58         OPER_INDEX  DB      ?       ; Operand Index
  0006                =2    59         OPERAND     DB      4 DUP(?); Operands
  ----                =2    60       ESCP_STRUC    ENDS
                      =2    61
  ----                =2    62     H19_MODE_STRUC  STRUC
  0000                =2    63         ALTERNATE   DB      ?       ; Alternate Key-pad Mode
  0001                =2    64         ANSI        DB      ?       ; ANSI Mode
  0002                =2    65         AUTO_CR     DB      ?       ; Auto Carriage-Return on Line-Feed
  0003                =2    66         AUTO_LF     DB      ?       ; Auto Line-Feed on Carriage-Return
  0004                =2    67         AUTO_REPEAT DB      ?       ; Auto-Repeat Keyboard
```

```
LOC  OBJ                  LINE     SOURCE

                     =1    123
                     =1    124      %IF(%NES(MTR100,%SEGMENT_LABEL)) THEN (
                     =1             ; INTVEC          SEGMENT PUBLIC 'DATA'
                     =1                               EXTRN   ITV_SST:DWORD
                     =1                               EXTRN   ITV_OBI:DWORD
                     =1             ;                 EXTRN   DS_PTR_O:WORD
                     =1             ;                 EXTRN   DS_PTR_S:WORD
                     =1             ; INTVEC          ENDS
                     =1               MONENT          SEGMENT PARA PUBLIC
                     =1                               EXTRN   MTR_RES:FAR     ; Monitor Reset
                     =1                               EXTRN   MTR_MON:FAR     ; Monitor CALL Entry
                     =1                               EXTRN   MTR_SWIM:FAR    ; Monitor Software Interrupt Entry
                     =1                               EXTRN   MTR_DCRT:FAR    ; Monitor Dumb  Terminal Emulator
                     =1                               EXTRN   MTR_SCRT:FAR    ; Monitor Smart Terminal Emulator
                     =1                               EXTRN   MTR_DKBD:FAR    ; Monitor Dumb  Keyboard Emulator
                     =1                               EXTRN   MTR_SKBD:FAR    ; Monitor Smart Keyboard Emulator
                     =1               MONENT          ENDS
                     =1               MTR100          SEGMENT PARA PUBLIC 'CODE'
                     =1                               %IF(%EXTENDED_MONITOR)THEN(
                     =1                                 EXTRN R8085:NEAR      ; Restore 8085
                     =1                                 )FI
                     =1                               EXTRN   SDS:NEAR        ; Set Data Segment
                     =1               MTR100          ENDS
                     =1               )FI
                     =1    145
                     =1    146
                     =1    147      %IF(%NES(MTR101,%SEGMENT_LABEL)) THEN (
                     =1               CODE            SEGMENT BYTE PUBLIC 'CODE'
                     =1                               EXTRN   MTR101:NEAR     ; PL/M-86 Monitor Loop
                     =1                               EXTRN   VIDEO_INTR:NEAR ; PL/M-86 Video Interrupt
                     =1                               EXTRN   D_CRT:NEAR      ; PL/M-86 Dumb Terminal
                     =1                               EXTRN   D_KBD:NEAR      ; PL/M-86 Dumb Key-Board
                     =1                               EXTRN   S_CRT:NEAR      ; PL/M-86 Smart Terminal
                     =1                               EXTRN   S_KBD:NEAR      ; PL/M-86 Smart Key-Board
                     =1                               EXTRN   TXMT:NEAR       ; PL/M-86 Transmit Screen Character
                     =1                               EXTRN   TTY_INTR:NEAR   ; PL/M-86 Terminal Interrupt
                     =1                               EXTRN   TTY_POLL:NEAR   ; PL/M-86 Interrupt Poll
                     =1
                     =1                               EXTRN   CRLF:NEAR       ; PL/M-86 CR-LF
                     =1                               EXTRN   INIT:NEAR       ; PL/M-86 Initialize Hardware
                     =1                               EXTRN   WRITEC:NEAR     ; PL/M-86 Write Character
                     =1               CODE            ENDS
                     =1               )FI
                     =1    164
                     =1    165
                     =1    166      %IF(%NES(DATA,%SEGMENT_LABEL)) THEN (
                     =1               DATA            SEGMENT WORD PUBLIC 'DATA'
                     =1                               EXTRN   HORZ_CHAR:BYTE  ; Column Index
                     =1                               EXTRN   VERT_LINE:BYTE  ; Row    Index
                     =1                               EXTRN   REGP:DWORD      ; Pointer to Saved Processor Registers
                     =1                               EXTRN   RESETF:BYTE     ; Hardware Reset Flag
                     =1
                     =1                               EXTRN   DCI:DWORD       ; Display Character Initialization
                     =1                               EXTRN   DFC:DWORD       ; Display Font Character
                     =1                               EXTRN   D_XMTC:DWORD    ; Dumb Terminal Transmit Character
```

```
LOC  OBJ                  LINE     SOURCE

                           181
                           182     %*DEFINE(DPB(name,count))(
                                   %name   DB      %count DUP(?)
                                           PUBLIC  %name)
                           183
                           184     %*DEFINE(DPD(name))(
                                   %name   DD      1 DUP(?)
                                           PUBLIC  %name)
                           185
                           186     %*DEFINE(DPS(name,sname))(
                                   %name   DB      SIZE(%sname) DUP(?)
                                           PUBLIC  %name)
                           187
                           188     %*DEFINE(DPW(name,count))(
                                   %name   DW      %count DUP(?)
                                           PUBLIC  %name)
                           189
                           190
                           191     $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           291     ;;;          not to move.                                                    ;;;
                           292     ;;;                                                                          ;;;
                           293     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                           294     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                           295
                           296
                           297
                           298
                           299     ;       Miscellaneous Hardware Latch Values
                           300
                           301     %DPB(BIIL,1)            ; Base Internal Interrupt Level
                           304     %DPB(BSIL,1)            ; Base S-100    Interrupt Level
                           307     %DPB(PSP,1)             ; Processor Swap Port
                           310     %DPD(REGP)              ; Pointer to Saved Processor Registers
                           313     %DPB(RESETF,1)          ; Reset Flag
                           316
                           317     ;       Terminal Emulator Variables
                           318
                           319     %DPS(COLOR,COLOR_STRUC)         ; Color Structure
                           322     %DPS(CRTC_CURSOR,CRTC_STRUC)    ; Cursor Start Address Structure
                           325     %DPS(CRTC_DISPLAY,CRTC_STRUC)   ; Display Start Address Structure
                           328     %DPS(ESCP,ESCP_STRUC)           ; Escape Processing Structure
                           331     %DPS(H19_MODE,H19_MODE_STRUC)   ; H-19 Mode Definitions
                           334     %DPS(H19_PAR,H19_PAR_STRUC)     ; H-19 Parameters
                           337     %DPS(XMT,XMT_STRUC)             ; Transmit Character Structure
                           340
                           341
                           342
                           343
------                     344     DATA    ENDS
                           345
                           346             END
```

```
NAME            TYPE     VALUE  ATTRIBUTES, XREFS

H19_MODE. . . . V BYTE   02B2H  DATA PUBLIC  332# 333
H19_MODE_STRUC. STRUC           SIZE=0012H  #FIELDS=17  62 80# 332
H19_PAR . . . . V BYTE   02C4H  DATA PUBLIC  335# 336
H19_PAR_STRUC . STRUC           SIZE=0009H  #FIELDS=9  82 92# 335
HORZ_CHAR . . . V BYTE   0291H  DATA PUBLIC  270# 271
INDEX . . . . . V BYTE   0000H  S FIELD  32#
INIT. . . . . L NEAR     0000H  EXTRN  160#
INSERT. . . . . V BYTE   000AH  S FIELD  73#
ITV_OBI . . . . V DWORD  0000H  EXTRN  127#
ITV_SST . . . . V DWORD  0000H  EXTRN  126#
KEY_EN. . . . . V BYTE   000BH  S FIELD  74#
KMAP. . . . . V BYTE   0091H  DATA PUBLIC  264# 265
LPS . . . . . V BYTE   0002H  S FIELD  85#
MDC . . . . . V DWORD  0073H  DATA PUBLIC  237# 238
MDL . . . . . V DWORD  0077H  DATA PUBLIC  240# 241
MODE. . . . . V BYTE   0003H  S FIELD  56#
MONENT. . . . . SEGMENT         SIZE=0000H PARA PUBLIC   131# 139
MSK . . . . . V BYTE   0002H  S FIELD  41#
MTR_DCRT. . . . L FAR    0000H  EXTRN  135#
MTR_DKBD. . . . L FAR    0000H  EXTRN  137#
MTR_FONT. . . . V BYTE   0000H  EXTRN  119#
MTR_MON . . . . L FAR    0000H  EXTRN  133#
MTR_RES . . . . L FAR    0000H  EXTRN  132#
MTR_SCRT. . . . L FAR    0000H  EXTRN  136#
MTR_SKBD. . . . L FAR    0000H  EXTRN  138#
MTR_SWIM. . . . L FAR    0000H  EXTRN  134#
MTR100. . . . . SEGMENT         SIZE=0000H PARA PUBLIC 'CODE'  140# 143 175
MTR101. . . . . L NEAR   0000H  EXTRN  149#
OPER_COUNT. . . V BYTE   0004H  S FIELD  57#
OPER_INDEX. . . V BYTE   0005H  S FIELD  58#
OPERAND . . . . V BYTE   0006H  S FIELD  59#
PAINTED . . . . V BYTE   0004H  S FIELD  43#
PFONT . . . . . V BYTE   0005H  S FIELD  44#
PLM_TRUE. . . . NUMBER   00FFH    20#
PORT. . . . . V BYTE   0001H  S FIELD  33#
POVRAM. . . . . V BYTE   0003H  S FIELD  86#
PROMPT. . . . . V DWORD  007BH  DATA PUBLIC  243# 244
PSP . . . . . V BYTE   0295H  DATA PUBLIC  308# 309
RDC . . . . . V DWORD  007FH  DATA PUBLIC  246# 247
REGP. . . . . V DWORD  0296H  DATA PUBLIC  311# 312
RESETF. . . . . V BYTE   029AH  DATA PUBLIC  314# 315
REVERSE . . . . V WORD   000CH  S FIELD  75#
ROW . . . . . V BYTE   0006H  S FIELD  99#
S_CRT . . . . . L NEAR   0000H  EXTRN  153#
S_KBD . . . . . L NEAR   0000H  EXTRN  154#
S_XMTC. . . . . V DWORD  0083H  DATA PUBLIC  249# 250
SDS . . . . . L NEAR   0000H  EXTRN  142#
SHIFTED . . . . V BYTE   000EH  S FIELD  76#
SLI . . . . . V BYTE   0004H  S FIELD  87#
SPC . . . . . V BYTE   0005H  S FIELD  88#
START . . . . . V WORD   0000H  S FIELD  49#
STATUS. . . . . V BYTE   000FH  S FIELD  77#
STRNG . . . . . V BYTE   0002H  S FIELD  34#
SW401 . . . . . V BYTE   0006H  S FIELD  89#
SW402 . . . . . V BYTE   0007H  S FIELD  90#
```

```
SERIES-III 8086/8087/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE MAIN
OBJECT MODULE PLACED IN :F2:MTR100.OBJ
(NVOCATION LINE CONTROLS:  PRINT(:TO:) XREF ERRORPRINT(:TO:)


LOC  OBJ                  LINE     SOURCE
```

```
LOC  OBJ                  LINE     SOURCE

                           27 +1   $ INCLUDE (:F2:I85DEF.COM)
                    =1     28      ;;      I85.DEF - Intel 8085/8080 Instruction Definitions
                    =1     29      ;
                    =1     30      ;           addr        - 16-bit Address
                    =1     31      ;           data        - 8-bit Data
                    =1     32      ;           port        - 8-bit Port Address
                    =1     33      ;
                    =1     34
  00F3              =1     35      I85_DI          EQU     363Q    ; DI            - Disable Interrupts
  00FB              =1     36      I85_EI          EQU     373Q    ; EI            - Enable  Interrupts
  00C3              =1     37      I85_JMP         EQU     303Q    ; JMP   addr    - Jump to Address
  002A              =1     38      I85_LHLD        EQU     052Q    ; LHLD  addr    - Load HL Direct
  003E              =1     39      I85_MVIA        EQU     076Q    ; MVI   A,data  - Set A to data
  00D3              =1     40      I85_OUT         EQU     323Q    ; OUT   port    - Output A to port
  00F1              =1     41      I85_POP_AF      EQU     361Q    ; POP   PSW     - Pop Processor Status Word
  00C1              =1     42      I85_POP_BC      EQU     301Q    ; POP   BC      - Pop BC Register Pair
  00D1              =1     43      I85_POP_DE      EQU     321Q    ; POP   DE      - Pop DE Register Pair
  00E1              =1     44      I85_POP_HL      EQU     341Q    ; POP   HL      - Pop HL Register Pair
  00F5              =1     45      I85_PUSH_AF     EQU     365Q    ; PUSH  PSW     - Push Processor Status Word
  00C9              =1     46      I85_RET         EQU     311Q    ; RET           - Return from Procedure
  0022              =1     47      I85_SHLD        EQU     042Q    ; SHLD  addr    - Store HL Direct
  00F9              =1     48      I85_SPHL        EQU     371Q    ; SPHL          - SP = HL
                           49 +1   $ INCLUDE (:F2:IODEF.COM)
                    =1     50      ;;      IODEF.COM
                    =1     51      ;
                    =1     52      ;       'IODEF.COM' contains the few I/O constants required
                    =1     53      ;       by assembly language routines.
                    =1     54      ;
                    =1     55
  00FF              =1     56      IO_DIP          EQU     00FFH   ; DIP-Switch Port
  00F6              =1     57      IO_DIAG         EQU     00F6H   ; Diagnostic Board Switch
  00FE              =1     58      IO_SWAP         EQU     00FEH   ; SWAP Port
                    =1     59
                    =1     60      ;       Misc.
                    =1     61
  0008              =1     62      DIP_AUTO_BOOT   EQU     00001000B       ; Auto-Boot Dip Switch
                           63 +1   $ INCLUDE (:F2:PARDEF.COM)
                    =1     64 +1   $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                     =1    107      ;;      Interrupt Usage
                     =1    108      ;
                     =1    109      ;       This common deck defines the MTR-100 Interrupt
                     =1    110      ;       usage.  Note that the interrupt vector for int-
                     =1    111      ;       errupt 255 is used to store a global data seg-
                     =1    112      ;       ment value.  This is somewhat undesirable to
                     =1    113      ;       reserve such a location, however, since all
                     =1    114      ;       other values are indexed off of it, it seems
                     =1    115      ;       alright.  The offset must be zero so that any
                     =1    116      ;       spurious interrupts will jump to the first addr-
                     =1    117      ;       ess of the data segment.  If these interrupts
                     =1    118      ;       are to be handled, the data segment vector may
                     =1    119      ;       be patched.
                     =1    120      ;
                     =1    121
-----                =1    122      INTVEC  SEGMENT PARA AT PAR_INT
                     =1    123
0000 (1              =1    124      ITV_DBZ DD      1 DUP(?)        ; Divide by zero interrupt
     ????????
     )
0004 (1              =1    125      ITV_SST DD      1 DUP(?)        ; Single Step
     ????????
     )
0008 (1              =1    126      ITV_NMI DD      1 DUP(?)        ; Non-Maskable Interrupt
     ????????
     )
000C (1              =1    127      ITV_OBI DD      1 DUP(?)        ; One Byte Instruction
     ????????
     )
0010 (1              =1    128      ITV_IOV DD      1 DUP(?)        ; Interrupt On Overflow
     ????????
     )
                     =1    129
0014 (1              =1    130      ITV_5   DD      1 DUP(?)        ; Interrupt 5
     ????????
     )
0018 (26             =1    131              DD      (32-6) DUP(?)   ; Reserved Interrupts
     ????????
     )
0080 (1              =1    132      ITV_32  DD      1 DUP(?)        ; Interrupt 32
     ????????
     )
0084 (222            =1    133              DD      (255-33) DUP(?) ; Allocate space for available user interrupts
     ????????
     )
03FC (1              =1    134      ITV_255 DD      1 DUP(?)        ; Interrupt 255 is used for MTR-100 DS
     ????????
     )
                     =1    135
  03FC               =1    136      MTR_DS_O    EQU    WORD PTR ITV_255+0   ; MTR-100 Data Segment Offset
                     =1    137                                             ;  (must be 0)
  03FE               =1    138      MTR_DS_S    EQU    WORD PTR ITV_255+2   ; MTR-100 Data Segment
                     =1    139
-----                =1    140      INTVEC  ENDS
                           141              PUBLIC  ITV_SST         ; Single Step
```

```
LOC  OBJ                  LINE     SOURCE

                           150     %*DEFINE(SEGMENT_LABEL)(MTR100)
                           151 +1  $INCLUDE(:F2:EXTERN.COM)
                     =1    152 +1  $SAVE
                     =1    153 +1  $NOGEN
                     =1    154
                     =1    155     %*DEFINE(EXTENDED_MONITOR)(0)    %' EXTENDED_MODE is FALSE
                     =1
  00FF               =1    156     PLM_TRUE        EQU     0FFH     ; PL/M-86 Boolean TRUE
                     =1    157
                     =1    158     ;       Structure Definitions
                     =1    159
                     =1    160     $       INCLUDE (':F2:STRDEF.COM')
                     =2    161     $SAVE
                     =2    162     $NOGEN
                     =2    163
                     =2    164     ;;      Global Structure Definitions
                     =2    165     ;
                     =2    166
  ------             =2    167     BOOT_PAR_STRUC  STRUC
  0000               =2    168         INDEX       DB      ?       ; Device Index
  0001               =2    169         PORT        DB      ?       ; Base Port Address
  0002               =2    170         STRNG       DB      80 DUP(?)       ; Parameter String Passed
  0052               =2    171         UNIT        DB      ?       ; Unit of Device Booted
  ------             =2    172       BOOT_PAR_STRUC        ENDS
                     =2    173
  ------             =2    174     COLOR_STRUC     STRUC
  0000               =2    175       FORE          DB      ?       ; Foreground Color
  0001               =2    176       BACK          DB      ?       ; Background Color
  0002               =2    177       MSK           DB      ?       ; SHL(Back,3) OR Fore
  0003               =2    178       CLEAR         DB      ?       ; Color to Clear
  0004               =2    179       PAINTED       DB      ?       ; Color to Fully Paint
  0005               =2    180       PFONT         DB      ?       ; Color to set to Font Pattern
  0006               =2    181       COMP_FONT     DB      ?       ; Color to set to Complement of Font Pattern
  ------             =2    182         COLOR_STRUC ENDS
                     =2    183
  ------             =2    184     CRTC_STRUC      STRUC
  0000               =2    185         START       DW      ?       ; CRT-C ..... Start Address
  0002               =2    186         UPDATE      DB      ?       ; True if update is required
  ------             =2    187       CRTC_STRUC    ENDS
                     =2    188
  ------             =2    189     ESCP_STRUC      STRUC
  0000               =2    190         COMMAND     DB      ?       ; Current Escape Command
  0001               =2    191         FUNCTION    DW      ?       ; Pointer to Escape Processor
  0003               =2    192         MODE        DB      ?       ; Escape Emulation Mode
  0004               =2    193         OPER_COUNT  DB      ?       ; Operand Count
  0005               =2    194         OPER_INDEX  DB      ?       ; Operand Index
  0006               =2    195         OPERAND     DB      4 DUP(?); Operands
  ------             =2    196       ESCP_STRUC    ENDS
                     =2    197
  ------             =2    198     H19_MODE_STRUC  STRUC
  0000               =2    199         ALTERNATE   DB      ?       ; Alternate Key-pad Mode
  0001               =2    200         ANSI        DB      ?       ; ANSI Mode
  0002               =2    201         AUTO_CR     DB      ?       ; Auto Carriage-Return on Line-Feed
  0003               =2    202         AUTO_LF     DB      ?       ; Auto Line-Feed on Carriage-Return
  0004               =2    203         AUTO_REPEAT DB      ?       ; Auto-Repeat Keyboard
```

```
LOC  OBJ                  LINE     SOURCE

                     =1    259
                     =1    260     %IF(%NES(MTR100,%SEGMENT_LABEL)) THEN (
                     =1            ; INTVEC        SEGMENT PUBLIC 'DATA'
                     =1                            EXTRN   ITV_SST:DWORD
                     =1                            EXTRN   ITV_OBI:DWORD
                     =1            ;               EXTRN   DS_PTR_O:WORD
                     =1            ;               EXTRN   DS_PTR_S:WORD
                     =1            ; INTVEC        ENDS
                     =1              MONENT        SEGMENT PARA PUBLIC
                     =1                            EXTRN   MTR_RES:FAR     ; Monitor Reset
                     =1                            EXTRN   MTR_MON:FAR     ; Monitor CALL Entry
                     =1                            EXTRN   MTR_SWIM:FAR    ; Monitor Software Interrupt Entry
                     =1                            EXTRN   MTR_DCRT:FAR    ; Monitor Dumb  Terminal Emulator
                     =1                            EXTRN   MTR_SCRT:FAR    ; Monitor Smart Terminal Emulator
                     =1                            EXTRN   MTR_DKBD:FAR    ; Monitor Dumb  Keyboard Emulator
                     =1                            EXTRN   MTR_SKBD:FAR    ; Monitor Smart Keyboard Emulator
                     =1              MONENT        ENDS
                     =1              MTR100        SEGMENT PARA PUBLIC 'CODE'
                     =1                            %IF(%EXTENDED_MONITOR)THEN(
                     =1                              EXTRN R8085:NEAR      ; Restore 8085
                     =1                              )FI
                     =1                            EXTRN   SDS:NEAR        ; Set Data Segment
                     =1              MTR100        ENDS
                     =1              )FI
                     =1    261
                     =1    262
                     =1    263     %IF(%NES(MTR101,%SEGMENT_LABEL)) THEN (
                     =1              CODE          SEGMENT BYTE PUBLIC 'CODE'
                     =1                            EXTRN   MTR101:NEAR     ; PL/M-86 Monitor Loop
                     =1                            EXTRN   VIDEO_INTR:NEAR ; PL/M-86 Video Interrupt
                     =1                            EXTRN   D_CRT:NEAR      ; PL/M-86 Dumb Terminal
                     =1                            EXTRN   D_KBD:NEAR      ; PL/M-86 Dumb Key-Board
                     =1                            EXTRN   S_CRT:NEAR      ; PL/M-86 Smart Terminal
                     =1                            EXTRN   S_KBD:NEAR      ; PL/M-86 Smart Key-Board
                     =1                            EXTRN   TXMT:NEAR       ; PL/M-86 Transmit Screen Character
                     =1                            EXTRN   TTY_INTR:NEAR   ; PL/M-86 Terminal Interrupt
                     =1                            EXTRN   TTY_POLL:NEAR   ; PL/M-86 Interrupt Poll
                     =1
                     =1                            EXTRN   CRLF:NEAR       ; PL/M-86 CR-LF
                     =1                            EXTRN   INIT:NEAR       ; PL/M-86 Initialize Hardware
                     =1                            EXTRN   WRITEC:NEAR     ; PL/M-86 Write Character
                     =1              CODE          ENDS
                     =1              )FI
                     =1    280
                     =1    281
                     =1    282     %IF(%NES(DATA,%SEGMENT_LABEL)) THEN (
                     =1              DATA          SEGMENT WORD PUBLIC 'DATA'
                     =1                            EXTRN   HORZ_CHAR:BYTE  ; Column Index
                     =1                            EXTRN   VERT_LINE:BYTE  ; Row    Index
                     =1                            EXTRN   REGP:DWORD      ; Pointer to Saved Processor Registers
                     =1                            EXTRN   RESETF:BYTE     ; Hardware Reset Flag
                     =1
                     =1                            EXTRN   DCI:DWORD       ; Display Character Initialization
                     =1                            EXTRN   DFC:DWORD       ; Display Font Character
                     =1                            EXTRN   D_XMTC:DWORD    ; Dumb Terminal Transmit Character
```

```
LOC  OBJ                  LINE     SOURCE

                           326     ;;;     Initial 8085 Boot Code
                           327     ;
                           328     ;       This code is initially invoked after reset.  Since
                           329     ;       the first processor to run is the 8085, this code
                           330     ;       switches to the 8086 processor.
                           331     ;
                           332     ;       Initially, the 8K-byte MTR-100 ROM is replicated
                           333     ;       throughout the memory space.  Since the 8085 resets
                           334     ;       location zero, this code must be the first in the
                           335     ;       ROM.  It promptly switches to the 8088, which it is
                           336     ;       assumed will turn off the ROM mapping.  The 8085 is
                           337     ;       left with the instruction pointer at 0.
                           338     ;
                           339
                           340
-----                      341     B8085   SEGMENT BYTE AT PAR_B85 ; Start the segment at base of ROM
                           342
0000 3E                    343             DB      I85_MVIA,80H    ; MVI   A,80H
0001 80
0002 C3                    344             DB      I85_JMP         ; JMP   0FFFEH
0003 FEFF                  345             DW      0FFFEH
                           346
-----                      347     B8085   ENDS
                           348 +1  $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           373     ;;;     MTR_ENT - Monitor Entry Points
                           374     ;
                           375     ;       These vectors are provided as alternate entry points
                           376     ;       into the monitor routines.
                           377     ;
                           378
-----                      379     MONENT  SEGMENT PARA PUBLIC             ; Monitor Entry Point Segment
                           380
                           381             ASSUME  NOTHING
                           382
0000                       383     MTR_RES         LABEL   FAR
                           384                     PUBLIC  MTR_RES
0000 EA0000------     R    385                     JMP FAR PTR MTR         ; Reset the monitor entry point
                           386
0005                       387     MTR_MON         LABEL   FAR
                           388                     PUBLIC  MTR_MON
0005 EA9C00----       R    389                     JMP FAR PTR MON         ; Monitor Entry Point
                           390
000A                       391     MTR_SWIM        LABEL   FAR
                           392                     PUBLIC  MTR_SWIM
000A EA2B00----       R    393                     JMP FAR PTR SWIM        ; Monitor Software Interrupt Entry
                           394
000F                       395     MTR_DCRT        LABEL   FAR
                           396                     PUBLIC  MTR_DCRT
000F EA8800----       R    397                     JMP FAR PTR DCRT        ; Dumb Terminal Processor
                           398
0014                       399     MTR_DKBD        LABEL   FAR
                           400                     PUBLIC  MTR_DKBD
0014 EA9200----       R    401                     JMP FAR PTR DKBD        ; Dumb Key-Board Processor
                           402
0019                       403     MTR_SCRT        LABEL   FAR
                           404                     PUBLIC  MTR_SCRT
0019 EAB800----       R    405                     JMP FAR PTR SCRT        ; Smart Terminal Processor
                           406
001E                       407     MTR_SKBD        LABEL   FAR
                           408                     PUBLIC  MTR_SKBD
001E EAAE00----       R    409                     JMP FAR PTR SKBD        ; Smart Keyboard Processor
                           410
0023                       411     MTR_TTY_INTR    LABEL   FAR
                           412                     PUBLIC  MTR_TTY_INTR
0023 EAC200----       R    413                     JMP FAR PTR TTYINTR     ; Terminal Interrupt Handler
                           414
0028                       415     MTR_TTY_POLL    LABEL   FAR
                           416                     PUBLIC  MTR_TTY_POLL
0028 EA7E00----       R    417                     JMP FAR PTR TTYPOLL     ; Terminal Poll Function
                           418
002D                       419     MTR_IRET        LABEL   FAR
                           420                     PUBLIC  MTR_IRET
002D EACB00----       R    421                     JMP FAR PTR MIRET       ; Interrupt Return
                           422
------                     423     MONENT  ENDS
                           424 +1  $ EJECT
```

```
LOC  OBJ                  LINE      SOURCE

                           467      ;;      SWIM    - Software Interrupt Monitor
                           468      ;
                           469      ;       SWIM is the software interrupt monitor entry point.  It
                           470      ;       is presumed that both the flags and return address are
                           471      ;       on the top of the stack.  Furthermore, it is assumed that
                           472      ;       the return address is a far return, ie. that both the code
                           473      ;       segment and instruction pointer are on the stack.
                           474      ;
                           475      ;       Entry:  (SP+4)  = Original Flags
                           476      ;               (SP+2)  = Original Code Segment
                           477      ;               (SP+0)  = Original Instruction Pointer
                           478      ;               All other registers as initialized
                           479
002B 50                    480      SWIM:   PUSH    AX
002C 53                    481              PUSH    BX
002D 51                    482              PUSH    CX
002E 52                    483              PUSH    DX
002F 8BC4                  484              MOV     AX,SP
0031 050E00                485              ADD     AX,(4+3)*2      ; Account for all parameters on stack
0034 50                    486              PUSH    AX
0035 55                    487              PUSH    BP
0036 56                    488              PUSH    SI
0037 57                    489              PUSH    DI
                           490
                           491      ;       PUSH    CS              ; CS is at bottom of stack
0038 1E                    492              PUSH    DS
0039 16                    493              PUSH    SS
003A 06                    494              PUSH    ES
                           495
003B E86400                496              CALL    SDS             ; Set Data Segment
003E 89260000        E     497              MOV     WORD PTR DS:REGP,SP     ; Save Pointer to Registers
0042 8C160200        E     498              MOV     WORD PTR DS:REGP+2,SS
                           499
0046 E80000          E     500      SWIM1:  CALL    MTR101          ; CALL MTR101
0049 EBFB                  501              JMP     SWIM1           ; Continue processing monitor commands
                           502 +1   $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           547     ;;      BDIAG   - Boot Diagnostics
                           548     ;
                           549     ;       BDIAG is performs any power-on diagnostics.
                           550     ;
                           551
0087                       552     BDIAG   PROC    NEAR
                           553
0087 C3                    554             RET
                           555
                           556     BDIAG   ENDP
                           557 +1  $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           583     ;;      DKBD    - Dumb Key-Board
                           584     ;
                           585     ;       DKBD is the assembly language entry point to read
                           586     ;       and translate a character from the keyboard.  This
                           587     ;       entry point is for 'dumb', ie. naive processing.
                           588     ;       Each keyboard character returns a unique entry.
                           589     ;
                           590     ;       Entry:  NONE
                           591     ;
                           592     ;       Exit:   AL      = Key-Board Value for character
                           593     ;
                           594     ;       Uses:   ???
                           595     ;
                           596
0092                       597     DKBD    PROC    FAR
0092 1E                    598             PUSH    DS
0093 E80C00                599             CALL    SDS             ; Set Data Segment
                           600
0096 50                    601             PUSH    AX
0097 E80000          E     602             CALL    D_KBD
                           603
009A 1F                    604             POP     DS
009B CB                    605             RET
                           606
                           607     DKBD    ENDP
                           608 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           661     ;;      TTYPOLL - Terminal Poll
                           662     ;
                           663     ;       TTYPOLL is a far call to the TTY_POLL routine.
                           664     ;
                           665     ;       Entry:  NONE
                           666     ;
                           667     ;       Exit:   AX      = TRUE if updates will occur
                           668     ;                         at the next vertical retrace
                           669     ;                       = FALSE otherwise
                           670     ;
                           671
009E                       672     TTYPOLL PROC    NEAR
                           673             PUBLIC  TTYPOLL
009E E80000         E      674             CALL    TTY_POLL
00A1 C3                    675             RET
                           676     TTYPOLL ENDP
                           677 +1  $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           733 +1  $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           761     ;;      SKBD    - Smart Keyboard
                           762     ;
                           763     ;       SKBD is a routine to return H19 compatible key codes
                           764     ;       from the keyboard.  This entails emulating the key-
                           765     ;       board modes and escape sequence generation.
                           766     ;
                           767     ;       Entry:  NONE
                           768     ;
                           769     ;       Exit:   AL      = Key-Board Value for character
                           770     ;
                           771     ;       Uses:   ???
                           772     ;
                           773
00AE                       774     SKBD    PROC    FAR
00AE 1E                    775             PUSH    DS
00AF E8F0FF                776             CALL    SDS             ; Set Data Segment
                           777
00B2 50                    778             PUSH    AX
00B3 E80000         E      779             CALL    S_KBD
                           780
00B6 1F                    781             POP     DS
00B7 CB                    782             RET
                           783
                           784     SKBD    ENDP
                           785 +1  $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                          816
                          817
                          818
                          819
                          820      ;;       TTYINTR - Terminal Interrupt
                          821      ;
                          822      ;        TTYINTR is the global entry vector for the Terminal Interrupt
                          823      ;        processor.  This routine saves all registers, sets the correct
                          824      ;        Data Segment, and invokes TTY_INTR for the interrupt processing
                          825      ;
                          826      ;        Entry:   NONE
                          827      ;
                          828      ;        Exit:    NONE
                          829      ;
                          830      ;        Uses:    NONE
                          831      ;
                          832
00C2                      833      TTYINTR PROC    FAR
                          834
                          835      %IF(%EXTENDED_MONITOR
                          836 +1       0)THEN(
                          837       PUSH AX
                          838       PUSH BX
                          839       PUSH CX
                          840       PUSH DX
                          841      ; PUSH SP
                          842       PUSH BP
                          843       PUSH SI
                          844       PUSH DI
                          845       PUSHF
                          846      ; PUSH CS
                          847      ; PUSH DS
                          848       PUSH SS
                          849       PUSH ES
                          850      )FI
                          851 +1
                          852
00C2 1E                   853              PUSH    DS
00C3 E8DCFF               854              CALL    SDS                ; Set the Data Segment
00C6 E80000         E     855              CALL    TTY_INTR
00C9 1F                   856              POP     DS
                          857
                          858      %IF(%EXTENDED_MONITOR
                          859 +1       0)THEN(
                          860       POP ES
                          861       POP SS
                          862      ; POP DS
                          863      ; POP CS
                          864       POPF
                          865       POP DI
                          866       POP SI
                          867       POP BP
                          868      ; POP SP
                          869       POP DX
                          870       POP CX
```

```
LOC  OBJ                  LINE     SOURCE

                          884
                          885
                          886
                          887
                          888     ;;      MIRET   -- Monitor Interrupt Return
                          889     ;
                          890     ;       MIRET is simply a vector pointing to an Interrupt
                          891     ;       Return instruction.
                          892     ;
                          893     ;       Entry:  Stack clean except for Interrupt Data, ie.
                          894     ;               flags, CS, and IP.
                          895     ;
                          896     ;       Exit:   From interrupt
                          897     ;
                          898
00CB                      899     MIRET   PROC    FAR
                          900
00CB CF                   901             IRET                    ; Return from interrupt
                          902
                          903     MIRET   ENDP
                          904
                          905
                          906
                          907
                          908     MTR100  ENDS
                          909
                          910
                          911
                          912
                          913             END
```

```
NAME              TYPE     VALUE  ATTRIBUTES, XREFS

TTY_POLL. . . .   L NEAR   0000H  EXTRN  273# 674
TTYINTR . . . .   L FAR    00C2H  MTR100  413 833# 878
TTYPOLL . . . .   L NEAR   009EH  MTR100 PUBLIC  417 672# 673 676
TXMT. . . . . .   L NEAR   0000H  EXTRN  271#
UIES. . . . . .   V DWORD  0000H  EXTRN  300#
UNIT. . . . . .   V BYTE   0052H  S FIELD  171#
UPDATE. . . . .   V BYTE   0002H  S FIELD  186#
UPDN. . . . . .   V BYTE   0010H  S FIELD  214#
VERT_LINE . . .   V BYTE   0000H  EXTRN  285#
VIDEO_INTR. . .   L NEAR   0000H  EXTRN  266#
VIDEO_INTR_A. .   L FAR    0000H  EXTRN  248#
VIDEOA. . . . .   SEGMENT         SIZE=0000H BYTE  PUBLIC 'CODE'  316# 317 321
VRAM_SIZE . . .   V BYTE   0008H  S FIELD  227#
WRAP. . . . . .   V BYTE   0011H  S FIELD  215#
WRITEC. . . . .   L NEAR   0000H  EXTRN  277#
XCA . . . . . .   V DWORD  0000H  EXTRN  301#
XMT . . . . . .   V TBYTE  0000H  EXTRN  310#
XMT_COLOR . . .   V BYTE   0007H  S FIELD  236#
XMT_GRAPHIC . .   V BYTE   0008H  S FIELD  237#
XMT_REVERSE . .   V BYTE   0009H  S FIELD  238#
XMT_STRUC . . .   STRUC            SIZE=000AH  #FIELDS=8  230 239# 310

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

```
SERIES-III 8086/8087/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE ASTLIB
OBJECT MODULE PLACED IN :F2:ASTLIB.OBJ
INVOCATION LINE CONTROLS:  PRINT(:TO:) XREF ERRORPRINT(:TO:)


LOC  OBJ                  LINE     SOURCE

                             1 +1  $TITLE('MTR100 Z-Machine Monitor ROM:  Assist Library')
                             2 +1  $GEN
                             3 +1  $SYMBOLS
                             4 +1  $XREF
                             5     ;;;     MTR-100 Assembly Language Assist Library
                             6     ;
                             7     ;       This library provides the assembly language assistance
                             8     ;       required by the PL/M-86 code in MTR-101.  These routines
                             9     ;       are meant to be called from PL/M-86, and assume that the
                            10     ;       model of computation is COMPACT.
                            11     ;
                            12     ;       By      Gregg Chandler
                            13     ;       Date    1-Jan-1982
                            14     ;
                            15     ;       Copyright(C) 1982, Heath Co.
                            16     ;
                            17
                            18     %*DEFINE(SEGMENT_LABEL)(ASTLIB)
                            19 +1  $INCLUDE(:F2:EXTERN.COM)
                  =1        20 +1  $SAVE
                  =1        21 +1  $NOGEN
                  =1        22
                  =1        23     %*DEFINE(EXTENDED_MONITOR)(0)   %' EXTENDED_MODE is FALSE
                  =1
  00FF            =1        24     PLM_TRUE        EQU     0FFH    ; PL/M-86 Boolean TRUE
                  =1        25
                  =1        26     ;       Structure Definitions
                  =1        27
                  =1        28     $       INCLUDE (':F2:STRDEF.COM')
                  =2        29     $SAVE
                  =2        30     $NOGEN
                  =2        31
                  =2        32     ;;      Global Structure Definitions
                  =2        33     ;
                  =2        34
------            =2        35     BOOT_PAR_STRUC  STRUC
0000              =2        36         INDEX       DB      ?       ; Device Index
0001              =2        37         PORT        DB      ?       ; Base Port Address
0002              =2        38         STRNG       DB      80 DUP(?)       ; Parameter String Passed
0052              =2        39         UNIT        DB      ?       ; Unit of Device Booted
------            =2        40       BOOT_PAR_STRUC        ENDS
                  =2        41
------            =2        42     COLOR_STRUC     STRUC
0000              =2        43       FORE          DB      ?       ; Foreground Color
0001              =2        44       BACK          DB      ?       ; Background Color
0002              =2        45       MSK           DB      ?       ; SHL(Back,3) OR Fore
0003              =2        46       CLEAR         DB      ?       ; Color to Clear
0004              =2        47       PAINTED       DB      ?       ; Color to Fully Paint
0005              =2        48       PFONT         DB      ?       ; Color to set to Font Pattern
0006              =2        49       COMP_FONT     DB      ?       ; Color to set to Complement of Font Pattern
```

```
LOC  OBJ                  LINE     SOURCE

0008                   =2   105          XMT_GRAPHIC DB      ?       ; Current GRAPHIC State Transmitted
0009                   =2   106          XMT_REVERSE DB      ?       ; Current REVERSE State Transmitted
-----                  =2   107        XMT_STRUC     ENDS
                       =2   108
                       =2   109
                       =2   110      $RESTORE
                       =1   111
                       =1   112      ;       Globals Definitions
                       =1   113
                       =1   114      %IF(%NES(ASTLIB,%SEGMENT_LABEL)) THEN (
                       =1             ASTLIB        SEGMENT BYTE PUBLIC 'CODE'
                       =1                           EXTRN   VIDEO_INTR_A:FAR
                       =1             ASTLIB        ENDS
                       =1             )FI
                       =1   115
                       =1   116
                       =1   117      %IF(%NES(FONTAB,%SEGMENT_LABEL)) THEN (
                       =1             FONTAB        SEGMENT BYTE PUBLIC 'DATA'
                       =1                           EXTRN   MTR_FONT:BYTE
                       =1             FONTAB        ENDS
                       =1             )FI
-----                  =1   122
                       =1   123
                       =1   124      %IF(%NES(MTR100,%SEGMENT_LABEL)) THEN (
                       =1             ; INTVEC      SEGMENT PUBLIC 'DATA'
                       =1                           EXTRN   ITV_SST:DWORD
                       =1                           EXTRN   ITV_OBI:DWORD
                       =1             ;             EXTRN   DS_PTR_O:WORD
                       =1             ;             EXTRN   DS_PTR_S:WORD
                       =1             ; INTVEC      ENDS
                       =1             MONENT        SEGMENT PARA PUBLIC
                       =1                           EXTRN   MTR_RES:FAR     ; Monitor Reset
                       =1                           EXTRN   MTR_MON:FAR     ; Monitor CALL Entry
                       =1                           EXTRN   MTR_SWIM:FAR    ; Monitor Software Interrupt Entry
                       =1                           EXTRN   MTR_DCRT:FAR    ; Monitor Dumb  Terminal Emulator
                       =1                           EXTRN   MTR_SCRT:FAR    ; Monitor Smart Terminal Emulator
                       =1                           EXTRN   MTR_DKBD:FAR    ; Monitor Dumb  Keyboard Emulator
                       =1                           EXTRN   MTR_SKBD:FAR    ; Monitor Smart Keyboard Emulator
                       =1             MONENT        ENDS
                       =1             MTR100        SEGMENT PARA PUBLIC 'CODE'
                       =1                           %IF(%EXTENDED_MONITOR)THEN(
                       =1                             EXTRN R8085:NEAR      ; Restore 8085
                       =1                             )FI
                       =1                           EXTRN   SDS:NEAR        ; Set Data Segment
                       =1             MTR100        ENDS
                       =1             )FI
-----                  =1   145
                       =1   146
                       =1   147      %IF(%NES(MTR101,%SEGMENT_LABEL)) THEN (
                       =1             CODE          SEGMENT BYTE PUBLIC 'CODE'
                       =1                           EXTRN   MTR101:NEAR     ; PL/M-86 Monitor Loop
                       =1                           EXTRN   VIDEO_INTR:NEAR ; PL/M-86 Video Interrupt
                       =1                           EXTRN   D_CRT:NEAR      ; PL/M-86 Dumb Terminal
                       =1                           EXTRN   D_KBD:NEAR      ; PL/M-86 Dumb Key-Board
                       =1                           EXTRN   S_CRT:NEAR      ; PL/M-86 Smart Terminal
```

```
LOC  OBJ                  LINE      SOURCE

                          209
                          210       ;       INCLUDE (':F2:PARDEF.COM')
                          211       ;       INCLUDE (':F2:INTDEF.COM')
                          212
                          213
------                    214       REG     STRUC
                          215
0000                      216         ES_   DW      ?               ; Extra Segment
0002                      217         SS_   DW      ?               ; Stack Segment
0004                      218         DS_   DW      ?               ; Data  Segment
0006                      219         DI_   DW      ?               ; Destination  Index
0008                      220         SI_   DW      ?               ; Source       Index
000A                      221         BP_   DW      ?               ; Base   Pointer
000C                      222         SP_   DW      ?               ; Stack  Pointer
000E                      223         DX_   DW      ?               ; Data
0010                      224         CX_   DW      ?               ; Count
0012                      225         BX_   DW      ?               ; Base
0014                      226         AX_   DW      ?               ; Accumulator
0016                      227         IP_   DW      ?               ; Instruction  Pointer
0018                      228         CS_   DW      ?               ; Code   Segment
001A                      229         FLAG_ DW      ?               ; Flags
                          230
------                    231       REG     ENDS
                          232
                          233
                          234               NAME    ASTLIB                  ; Name the module
                          235
------                    236       ASTLIB  SEGMENT BYTE PUBLIC 'CODE'      ; Concatenate with the Code Segment
                          237               ASSUME  CS:CGROUP,DS:DATA       ; Assume MTR-100 Segments
                          238 +1    $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           281     ;;      FLAGS   -- Return 8086 Flags
                           282     ;
                           283     ;       FLAGS returns the current 8086 flag settings
                           284     ;       as a WORD.
                           285     ;
                           286     ;       Entry:  NONE
                           287     ;
                           288     ;       Exit:   AX      = Entry flags
                           289     ;
                           290
000E                       291     FLAGS   PROC    NEAR
                           292             PUBLIC  FLAGS
000E 9C                    293             PUSHF
000F 58                    294             POP     AX              ; AX = Flags
0010 C3                    295             RET
                           296     FLAGS   ENDP
                           297 +1  $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                          341
                          342
                          343
                          344
                          345      ;;       INCB    - Increment Byte
                          346      ;
                          347      ;        INCB increments the specified byte.  The primary
                          348      ;        use for this routine is to force a value to be
                          349      ;        updated, rather than letting the compiler optimize
                          350      ;        variable references to said variable.
                          351      ;
                          352      ;        USAGE:           CALL    INCB(byte$ptr);
                          353      ;
                          354      ;        Entry:           byte$ptr        - Address of byte to increment
                          355      ;
                          356
------                    357      INCB_P           STRUC
0000                      358                       DW      ?
0002                      359                       DW      ?
0004                      360        INCB_BYTEPTR   DD      ?
------                    361          INCB_P       ENDS
                          362
                          363
0011                      364      INCB     PROC    NEAR
                          365               PUBLIC  INCB
0011 55                   366               PUSH    BP
0012 8BEC                 367               MOV     BP,SP
                          368
0014 C47E04               369               LES     DI,[BP].INCB_BYTEPTR
0017 26FE05               370               INC     BYTE PTR ES:[DI]
                          371
001A 5D                   372               POP     BP
001B C20400               373               RET     4
                          374      INCB     ENDP
                          375 +1   $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                          423      ;;      SXMTC   - Smart Terminal Transmit Character
                          424      ;
                          425      ;       'SXMTC' Is invoked by the smart keyboard/terminal to
                          426      ;       transmit generated characters.  This routine merely
                          427      ;       transforms the PL/M-86 calling convention into a more
                          428      ;       conventional assembly language one by moving the sup-
                          429      ;       plied byte from the stack to the AL register, and
                          430      ;       then invokes the routine pointed to by 'S_XMTC'.
                          431      ;
                          432      ;
                          433      ;       Usage:          CALL S$XMTC(c);
                          434      ;
                          435      ;       Parameters:
                          436      ;
                          437      ;           c   - Character generated by the emulator
                          438      ;
                          439      ;       Exit:   NONE
                          440      ;
                          441
                          442
-----                     443      SXMTCP          STRUC
0000                      444                      DW      ?
0002                      445                      DW      ?
0004                      446       SXMTC_C        DB      ?
0005                      447                      DB      ?
-----                     448        SXMTCP        ENDS
                          449
                          450
001E                      451      SXMTC   PROC    NEAR
                          452              PUBLIC  SXMTC
001E 55                   453              PUSH    BP
001F 8BEC                 454              MOV     BP,SP
                          455
0021 8A4604               456              MOV     AL,[BP].SXMTC_C
0024 FF1E0000       E     457              CALL    DS:S_XMTC               ; Invoke vectored routine
                          458
0028 5D                   459              POP     BP
0029 C20200               460              RET     2
                          461      SXMTC   ENDP
                          462 +1   $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           515
                           516
                           517
                           518     ;;       XEC     - Execute
                           519     ;
                           520     ;        XEC is called to begin execution at the specified address.
                           521     ;        This routine assumes the registers are saved in the order
                           522     ;        specified in SWIM in the MTR100 module.  It is necessary
                           523     ;        to play some tricks to account for programs which may have
                           524     ;        modified the Stack Segment.
                           525     ;
                           526     ;        Usage:  CALL    XEC(reg$p)
                           527     ;
                           528     ;        Entry:  reg$p   = Base Address of register structure    POINTER
                           529     ;
                           530     ;        Exit:   to new address
                           531     ;
                           532
------                     533     XECP    STRUC
0000                       534                     DW      ?
0002                       535                     DW      ?
0004                       536      XECP_REGP      DD      ? ; Pointer to register structure
------                     537      XECP   ENDS
                           538
                           539
0042                       540     XEC     PROC    NEAR
                           541             PUBLIC  XEC
0042 55                    542             PUSH    BP
0043 8BEC                  543             MOV     BP,SP
                           544
                           545     ;       Initialize Software Interrupt Vectors
                           546
                           547     %IF(%EXTENDED_MONITOR
                           548 +1      0)THEN(
                           549      MOV AX,SEG ITV_SST
                           550      MOV ES,AX  ; ES = Interrupt Segment
                           551      MOV AX,OFFSET MTR_SWIM
                           552      MOV WORD PTR ES:ITV_SST+0,AX
                           553      MOV WORD PTR ES:ITV_OBI+0,AX
                           554      MOV AX,SEG MTR_SWIM
                           555      MOV WORD PTR ES:ITV_SST+2,AX
                           556      MOV WORD PTR ES:ITV_OBI+2,AX
                           557      )FI
                           558 +1
                           559
                           560     ;       Restore Saved Registers
                           561
0045 C57604                562             LDS     SI,[BP].XECP_REGP       ; DS:SI = Register Structure
                           563
                           564     %IF(%EXTENDED_MONITOR
                           565 +1      0) THEN (
                           566      MOV DI,DS:[SI].SP_ ; DI = New Stack Pointer
                           567      SUB DI,SIZE REG ; DI = New Structure Offset
                           568      MOV ES,DS:[SI].SS_ ; ES = New Stack Base Address
                           569      MOV CX,SIZE REG ; CX = Number of Bytes to copy
```

```
LOC  OBJ                  LINE     SOURCE

                           604
                           605
                           606
                           607
                           608     ;;      INIT_ITV        - Initialize Interrupt Vectors
                           609     ;
                           610     ;       INIT_ITV initializes the specified interrupt vector
                           611     ;       to the specified pointer value.
                           612     ;
                           613     ;       Usage:          CALL INIT_ITV(ivi,ivp);
                           614     ;
                           615     ;       Parameters
                           616     ;
                           617     ;           ivi = Interrupt Vector Index                (BYTE)
                           618     ;           ivp = Pointer to Interrupt Vector Routine   (POINTER)
                           619     ;
                           620     ;       Exit:   Specified Interrupt Vector Initialized to specified
                           621     ;               pointer value.
                           622     ;
                           623
------                     624     INIT_ITVP       STRUC
0000                       625                     DW      ?
0002                       626                     DW      ?
0004                       627      IVP            DD      ?
0008                       628      IVI            DB      ?
0009                       629                     DB      ?               ; An entire word is on the stack
------                     630       INIT_ITVP     ENDS
                           631
0061                       632     INIT_ITV        PROC    NEAR
                           633             PUBLIC  INIT_ITV
0061 55                    634             PUSH    BP
0062 8BEC                  635             MOV     BP,SP
0064 1E                    636             PUSH    DS
                           637
                           638     ;       Compute address of interrupt vector
                           639
0065 8A4608                640             MOV     AL,[BP].IVI
0068 02C0                  641             ADD     AL,AL
006A 02C0                  642             ADD     AL,AL
006C B400                  643             MOV     AH,0
006E 8BF8                  644             MOV     DI,AX           ; DI = 4 * IVI
                           645
0070 B8-----          E    646             MOV     AX,SEG ITV_SST
0073 8ED8                  647             MOV     DS,AX           ; ES    = Interrupt Vector Segment
0075 C44604                648             LES     AX,[BP].IVP     ; ES:AX = Pointer to routine
0078 8905                  649             MOV     [DI+0],AX       ; Set Offset
007A 8C4502                650             MOV     [DI+2],ES       ; Set Segment
                           651
007D 1F                    652             POP     DS
007E 5D                    653             POP     BP
007F C20600                654             RET     06H
                           655     INIT_ITV        ENDP
                           656
                           657
                           658
```

```
LOC  OBJ                  LINE     SOURCE

                           661
                           662
                           663
                           664
                           665     ;;      @MOVE   - Move
                           666     ;
                           667     ;       @MOVE moves the specified bytes from one location to
                           668     ;       another.  Though the 8086 has instructions tuned for
                           669     ;       this operation, it is necessary to first compute the
                           670     ;       direction of the move--whether from high to low, or
                           671     ;       low to high.  Since the true 20-bit addresses must be
                           672     ;       used for comparison, this takes a little work.
                           673     ;
                           674     ;
                           675     ;       NOTE:   This routine will not wrap out of
                           676     ;               any of the segment boundaries,
                           677     ;               rather, it wraps around within
                           678     ;               them.
                           679     ;
                           680     ;       Entry:  DS      = Source Segment
                           681     ;               SI      = Source Offset
                           682     ;               ES      = Destination Segment
                           683     ;               DI      = Destination Offset
                           684     ;               CX      = Byte Count
                           685     ;
                           686     ;       Exit:   Byte moved, SI and DI advanced, CX=0
                           687     ;
                           688     ;       Uses:   ALL
                           689     ;
                           690
                           691     %IF(%EXTENDED_MONITOR
                           692 +1      0) THEN (
                           693
                           694     @MOVE PROC NEAR
                           695
                           696      AND CX,CX
                           697      JNZ MOV0  ; There are bytes to move
                           698
                           699      RET   ; No bytes to move
                           700
                           701     MOV0: PUSH CX
                           702
                           703     ; Compute Absolute Source Address
                           704
                           705      MOV AX,DS
                           706      MOV DL,AH
                           707      MOV CL,4
                           708      SHL AX,CL
                           709      SHR DL,CL
                           710      ADD AX,SI
                           711      ADC DL,0  ; DL,AH,AL = Source Address
                           712
                           713     ; Compute Absolute Destination Address
                           714
                           715      MOV BX,ES
```

XREF SYMBOL TABLE LISTING

```
NAME             TYPE     VALUE  ATTRIBUTES, XREFS

??SEG . . . . . SEGMENT          SIZE=0000H PARA PUBLIC
ALTERNATE . . . V BYTE   0000H  S FIELD  67#
ANSI. . . . . . V BYTE   0001H  S FIELD  68#
ASTLIB. . . . . SEGMENT          SIZE=0082H BYTE  PUBLIC 'CODE'  205# 236 756
AUTO_CR . . . . V BYTE   0002H  S FIELD  69#
AUTO_LF . . . . V BYTE   0003H  S FIELD  70#
AUTO_REPEAT . . V BYTE   0004H  S FIELD  71#
AX_ . . . . . . V WORD   0014H  S FIELD  226#
BACK. . . . . . V BYTE   0001H  S FIELD  44#
BASE. . . . . . V WORD   0006H  S FIELD  316#
BCOUNT. . . . . V WORD   0001H  S FIELD  100#
BOOT_PAR. . . . V   83   0000H  EXTRN  187#
BOOT_PAR_STRUC. STRUC            SIZE=0053H  #FIELDS=4  35 40# 187
BP_ . . . . . . V WORD   000AH  S FIELD  221#
BURST . . . . . V BYTE   0000H  S FIELD  99#
BWO . . . . . . V BYTE   0005H  S FIELD  72#
BX_ . . . . . . V WORD   0012H  S FIELD  225#
CGROUP. . . . . GROUP            MTR100 CODE ASTLIB VIDEOA   205# 237
CLEAR . . . . . V BYTE   0003H  S FIELD  46#
CODE. . . . . . SEGMENT          SIZE=0000H BYTE  PUBLIC 'CODE'  148# 162 205
COL . . . . . . V BYTE   0005H  S FIELD  102#
COLOR . . . . . V     7  0000H  EXTRN  188#
COLOR_STRUC . . STRUC            SIZE=0007H  #FIELDS=7  42 50# 188
COMMAND . . . . V BYTE   0000H  S FIELD  58#
COMP_FONT . . . V BYTE   0006H  S FIELD  49#
COUNT . . . . . V WORD   0003H  S FIELD  101#
CPL . . . . . . V BYTE   0000H  S FIELD  87#
CRLF. . . . . . L NEAR   0000H  EXTRN  159#
CRTC_CURSOR . . V     3  0000H  EXTRN  189#
CRTC_DISPLAY. . V     3  0000H  EXTRN  190#
CRTC_STRUC. . . STRUC            SIZE=0003H  #FIELDS=2  52 55# 189 190
CS_ . . . . . . V WORD   0018H  S FIELD  228#
CURSOR. . . . . V BYTE   0006H  S FIELD  73#
CURSOR_ON . . . V BYTE   0007H  S FIELD  74#
CX_ . . . . . . V WORD   0010H  S FIELD  224#
D_CRT . . . . . L NEAR   0000H  EXTRN  151#
D_KBD . . . . . L NEAR   0000H  EXTRN  152#
D_XMTC. . . . . V DWORD  0000H  EXTRN  175# 274
DATA. . . . . . SEGMENT          SIZE=0000H WORD  PUBLIC 'DATA'  167# 195 237
DCI . . . . . . V DWORD  0000H  EXTRN  173#
DFC . . . . . . V DWORD  0000H  EXTRN  174#
DI_ . . . . . . V WORD   0006H  S FIELD  219#
DS_ . . . . . . V WORD   0004H  S FIELD  218#
DSC . . . . . . V BYTE   0001H  S FIELD  88#
DX_ . . . . . . V WORD   000EH  S FIELD  223#
DXMTC . . . . . L NEAR   0000H  ASTLIB PUBLIC  268# 269 278
DXMTC_C . . . . V BYTE   0004H  S FIELD  263# 273
DXMTCP. . . . . STRUC            SIZE=0006H  #FIELDS=4  260 265#
EDC . . . . . . V DWORD  0000H  EXTRN  176#
EMEC. . . . . . V DWORD  0000H  EXTRN  177#
ES_ . . . . . . V WORD   0000H  S FIELD  216#
```

```
NAME            TYPE     VALUE  ATTRIBUTES, XREFS

PROMPT. . . . . V DWORD  0000H  EXTRN  181#
RDC . . . . . . V DWORD  0000H  EXTRN  182#
REG . . . . . . STRUC           SIZE=001CH  #FIELDS=14  214 231# 578
REGP. . . . . . V DWORD  0000H  EXTRN  170#
RESETF. . . . . V BYTE   0000H  EXTRN  171#
REVERSE . . . . V WORD   000CH  S FIELD  79#
ROW . . . . . . V BYTE   0006H  S FIELD  103#
S_CRT . . . . . L NEAR   0000H  EXTRN  153#
S_KBD . . . . . L NEAR   0000H  EXTRN  154#
S_XMTC. . . . . V DWORD  0000H  EXTRN  183# 457
SDS . . . . . . L NEAR   0000H  EXTRN  142#
SHIFTED . . . . V BYTE   000EH  S FIELD  80#
SI_ . . . . . . V WORD   0008H  S FIELD  220#
SLI . . . . . . V BYTE   0004H  S FIELD  91#
SP_ . . . . . . V WORD   000CH  S FIELD  222# 577
SPC . . . . . . V BYTE   0005H  S FIELD  92#
SS_ . . . . . . V WORD   0002H  S FIELD  217# 576
START . . . . . V WORD   0000H  S FIELD  53#
STATUS. . . . . V BYTE   000FH  S FIELD  81#
STRNG . . . . . V BYTE   0002H  S FIELD  38#
SW401 . . . . . V BYTE   0006H  S FIELD  93#
SW402 . . . . . V BYTE   0007H  S FIELD  94#
SXMTC . . . . . L NEAR   001EH  ASTLIB PUBLIC  451# 452 461
SXMTC_C . . . . V BYTE   0004H  S FIELD  446# 456
SXMTCP. . . . . STRUC           SIZE=0006H  #FIELDS=4  443 448#
TTY_INTR. . . . L NEAR   0000H  EXTRN  156#
TTY_POLL. . . . L NEAR   0000H  EXTRN  157#
TXMT. . . . . . L NEAR   0000H  EXTRN  155#
UIES. . . . . . V DWORD  0000H  EXTRN  184#
UNIT. . . . . . V BYTE   0052H  S FIELD  39#
UPDATE. . . . . V BYTE   0002H  S FIELD  54#
UPDN. . . . . . V BYTE   0010H  S FIELD  82#
VAL . . . . . . V WORD   0008H  S FIELD  317#
VERT_LINE . . . V BYTE   0000H  EXTRN  169#
VIDEO_INTR. . . L NEAR   0000H  EXTRN  150# 492
VIDEO_INTR_A. . L FAR    002CH  ASTLIB PUBLIC  474# 475 509
VIDEOA. . . . . SEGMENT         SIZE=0000H BYTE  PUBLIC 'CODE'  200# 201 205
VRAM_SIZE . . . V BYTE   0008H  S FIELD  95#
WRAP. . . . . . V BYTE   0011H  S FIELD  83#
WRITEC. . . . . L NEAR   0000H  EXTRN  161#
XCA . . . . . . V DWORD  0000H  EXTRN  185#
XEC . . . . . . L NEAR   0042H  ASTLIB PUBLIC  540# 541 598
XECP. . . . . . STRUC           SIZE=0008H  #FIELDS=3  533 537#
XECP_REGP . . . V DWORD  0004H  S FIELD  536# 562
XMT . . . . . . V TBYTE  0000H  EXTRN  194#
XMT_COLOR . . . V BYTE   0007H  S FIELD  104#
XMT_GRAPHIC . . V BYTE   0008H  S FIELD  105#
XMT_REVERSE . . V BYTE   0009H  S FIELD  106#
XMT_STRUC . . . STRUC           SIZE=000AH  #FIELDS=8  98 107# 194

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

```
SERIES-III 8086/8087/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE VIDEOA
OBJECT MODULE PLACED IN :F2:VIDEOA.OBJ
INVOCATION LINE CONTROLS:  PRINT(:TO:) XREF ERRORPRINT(:TO:)


LOC  OBJ                  LINE     SOURCE

                             1 +1  $TITLE('MTR100 Z-Machine Monitor ROM:  Assembly Video Library')
                             2 +1  $GEN
                             3 +1  $SYMBOLS
                             4 +1  $XREF
                             5     ;;;     MTR100 Video Library Assembly Procedures
                             6     ;
                             7     ;       Copyright(C) 1982, by:
                             8     ;
                             9     ;               Heath Co.
                            10     ;               Benton Harbor, MI
                            11     ;                       49022
                            12     ;
                            13     ;
                            14     ;       This library implements the critical assemlby language
                            15     ;       routines necessary for the video library.  These routines
                            16     ;       assume the PL/M-86 calling convention and the COMPACT
                            17     ;       model of computation, however, they are invoked via "far"
                            18     ;       calls.
                            19     ;
                            20     ;       These routines assumes that the CRT-C is programmed for 9
                            21     ;       scan lines per character, and that the current VRMM par-
                            22     ;       ameters are valid.  The video address is computed as:
                            23     ;
                            24     ;               vert*16*128+char = vert*8*256+char
                            25     ;
                            26     ;
                            27     ;       By:     G. Chandler
                            28     ;       Date:   7-Jan-1982
                            29     ;
                            30
                            31 +1  $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                    =1      35     ;;      Paragraph Usage
                    =1      36     ;
                    =1      37     ;       Due to reset hardware conditions, various paragraphs
                    =1      38     ;       within the ROM are hard-coded to represent certain
                    =1      39     ;       values.  The current paragraph usage is as follows:
                    =1      40     ;
                    =1      41     ;               Paragraph       Usage
                    =1      42     ;               ---------       -----
                    =1      43     ;
                    =1      44     ;                 0000H         Interrupt Vectors
                    =1      45     ;                 0040H         User Program
                    =1      46     ;                  : :           :    :
                    =1      47     ;                 BFFFH         User Program
                    =1      48     ;                 C000H         Video Ram:  Blue  Plane
                    =1      49     ;                 D000H         Video Ram:  Red   Plane
                    =1      50     ;                 E000H         Video Ram:  Green Plane
                    =1      51     ;                 F000H         Reserved
                    =1      52     ;                  : :           :    :
                    =1      53     ;                 FDFFH         Reserved
                    =1      54     ;                 FE00H         MTR-100:  8085 Processor Switch Code
                    =1      55     ;                 FE01H         MTR-100:  MTR-100 Entry Vectors
                    =1      56     ;                 FE04H         MTR-100:  MTR-100 Code
                    =1      57     ;                  : :           :              :
                    =1      58     ;                  : :           :              :
                    =1      59     ;                 FFFFH         MTR-100:  Jump Far to MTR-100 Start
                    =1      60     ;
                    =1      61
0020                =1      62     SS_SIZE EQU     2*16            ; Paragraphs for initial Stack Segment
0040                =1      63     DS_SIZE EQU     4*16            ; Paragraphs for initial Data  Segment
                    =1      64
0000                =1      65     PAR_INT EQU     00000H          ; Interrupt Vectors
0040                =1      66     PAR_USR EQU     00040H          ; User Program
C000                =1      67     PAR_VID EQU     0C000H          ; Video RAM
C000                =1      68     PAR_BLU EQU     0C000H          ;   Blue  Plane
D000                =1      69     PAR_RED EQU     0D000H          ;   Red   Plane
E000                =1      70     PAR_GRN EQU     0E000H          ;   Green Plane
                    =1      71
FE00                =1      72     PAR_B85 EQU     0FE00H          ; 8085 Boot Paragraph
FE01                =1      73     PAR_MTR EQU     0FE01H          ; MTR-100 Entry Vectors
FFFF                =1      74     PAR_B86 EQU     0FFFFH          ; 8088 Boot Paragraph
                        75
                        76     %*DEFINE(SEGMENT_LABEL)(VIDEOA)
                        77 +1  $ INCLUDE (:F2:EXTERN.COM)
                    =1  78 +1  $SAVE
                    =1  79 +1  $NOGEN
                    =1  80
                    =1  81     %*DEFINE(EXTENDED_MONITOR)(0)    %' EXTENDED_MODE is FALSE
                    =1
00FF                =1  82     PLM_TRUE        EQU     0FFH    ; PL/M-86 Boolean TRUE
                    =1  83
                    =1  84     ;       Structure Definitions
                    =1  85
                    =1  86     $       INCLUDE (':F2:STRDEF.COM')
                    =2  87     $SAVE
                    =2  88     $NOGEN
```

```
LOC  OBJ                  LINE     SOURCE

                          =2   144     H19_PAR_STRUC   STRUC
0000                      =2   145         CPL         DB      ?       ; Characters Per Line
0001                      =2   146         DSC         DB      ?       ; Displayed Scan Lines per Character
0002                      =2   147         LPS         DB      ?       ; Lines Per Screen
0003                      =2   148         POVRAM      DB      ?       ; Planes of Video RAM
0004                      =2   149         SLI         DB      ?       ; Status Line Index
0005                      =2   150         SPC         DB      ?       ; Scan Lines per Character
0006                      =2   151         SW401       DB      ?       ; H19-Switch 401
0007                      =2   152         SW402       DB      ?       ; H19-Switch 402
0008                      =2   153         VRAM_SIZE   DB      ?       ; 0-32KBytes, 1-64KBytes
                          =2   154       H19_PAR_STRUC ENDS
                          =2   155
                          =2   156     XMT_STRUC       STRUC
0000                      =2   157         BURST       DB      ?       ; Characters to Transmit per VSYNC
0001                      =2   158         BCOUNT      DW      ?       ; Remaining Characters in current Burst
0003                      =2   159         COUNT       DW      ?       ; Characters left to Transmit
0005                      =2   160         COL         DB      ?       ; Horizontal Column to Transmit
0006                      =2   161         ROW         DB      ?       ; Vertical   Row    to Transmit
0007                      =2   162         XMT_COLOR   DB      ?       ; Current COLOR   State Transmitted
0008                      =2   163         XMT_GRAPHIC DB      ?       ; Current GRAPHIC State Transmitted
0009                      =2   164         XMT_REVERSE DB      ?       ; Current REVERSE State Transmitted
                          =2   165       XMT_STRUC     ENDS
                          =2   166
                          =2   167
                          =2   168     $RESTORE
                          =1   169
                          =1   170     ;       Globals Definitions
                          =1   171
                          =1   172     %IF(%NES(ASTLIB,%SEGMENT_LABEL)) THEN (
                          =1           ASTLIB          SEGMENT BYTE PUBLIC 'CODE'
                          =1                           EXTRN   VIDEO_INTR_A:FAR
                          =1           ASTLIB          ENDS
                          =1           )FI
                          =1   177
                          =1   178
                          =1   179     %IF(%NES(FONTAB,%SEGMENT_LABEL)) THEN (
                          =1           FONTAB          SEGMENT BYTE PUBLIC 'DATA'
                          =1                           EXTRN   MTR_FONT:BYTE
                          =1           FONTAB          ENDS
                          =1           )FI
                          =1   184
                          =1   185
                          =1   186     %IF(%NES(MTR100,%SEGMENT_LABEL)) THEN (
                          =1           ; INTVEC        SEGMENT PUBLIC 'DATA'
                          =1                           EXTRN   ITV_SST:DWORD
                          =1                           EXTRN   ITV_OBI:DWORD
                          =1           ;               EXTRN   DS_PTR_O:WORD
                          =1           ;               EXTRN   DS_PTR_S:WORD
                          =1           ; INTVEC        ENDS
                          =1           MONENT          SEGMENT PARA PUBLIC
                          =1                           EXTRN   MTR_RES:FAR     ; Monitor Reset
                          =1                           EXTRN   MTR_MON:FAR     ; Monitor CALL Entry
                          =1                           EXTRN   MTR_SWIM:FAR    ; Monitor Software Interrupt Entry
                          =1                           EXTRN   MTR_DCRT:FAR    ; Monitor Dumb  Terminal Emulator
                          =1                           EXTRN   MTR_SCRT:FAR    ; Monitor Smart Terminal Emulator
```

```
LOC  OBJ                  LINE      SOURCE

                     =1                           EXTRN   CRTC_DISPLAY:CRTC_STRUC
                     =1                           EXTRN   ESCP:ESCP_STRUC
                     =1                           EXTRN   H19_MODE:H19_MODE_STRUC
                     =1                           EXTRN   H19_PAR:H19_PAR_STRUC
                     =1                           EXTRN   XMT:XMT_STRUC
                     =1                   DATA    ENDS
                     =1                   )FI
------               =1        259
                     =1        260
                     =1        261       %IF(%NES(VIDEOA,%SEGMENT_LABEL)) THEN (
                     =1                   VIDEOA  SEGMENT BYTE PUBLIC 'CODE'
                     =1                   VIDEOA  ENDS
                     =1                   )FI
                     =1        262
                     =1        263
                     =1        264       CGROUP  GROUP   MTR100,CODE,ASTLIB,VIDEOA
                     =1        265
                     =1        266
                     =1        267       $RESTORE
                               268
                               269               NAME    VIDEOA
                               270
----                           271       VIDEOA  SEGMENT BYTE PUBLIC 'CODE'       ; Concatenate with the Code Segment
                               272               ASSUME  CS:CGROUP,DS:DATA
                               273
                               274
  000B                         275       SLPC    EQU     9+2     ; Scan Lines Per Character, 9 for char. + 2 for attrib.
                               276
  00D8                         277       VID_CMD EQU     0D8H            ; Video Command Port
                               278
                               279
                               280       ;       Color Bit Definitions in Video Command Port
                               281
  0000                         282       CB_BLK  EQU     000B                    ; Black
  0004                         283       CB_BLU  EQU     100B                    ; Blue
  0001                         284       CB_RED  EQU     001B                    ; Red
  0002                         285       CB_GRN  EQU     010B                    ; Green
                               286
  0005                         287       CB_MAG  EQU     CB_RED+CB_BLU           ; Magenta
  0006                         288       CB_CYN  EQU     CB_GRN+CB_BLU           ; Cyan
  0003                         289       CB_YEL  EQU     CB_GRN+CB_RED           ; Yellow
  0007                         290       CB_WHT  EQU     CB_GRN+CB_RED+CB_BLU    ; White
                               291
                               292
                               293
  0000                         294       CL_BLK  EQU     000B                    ; Black
  0001                         295       CL_BLU  EQU     001B                    ; Blue
  0002                         296       CL_RED  EQU     010B                    ; Red
  0003                         297       CL_MAG  EQU     CL_RED+CL_BLU           ; Magenta
  0004                         298       CL_GRN  EQU     100B                    ; Green
  0005                         299       CL_CYN  EQU     CL_GRN+CL_BLU           ; Cyan
  0006                         300       CL_YEL  EQU     CL_GRN+CL_RED           ; Yellow
  0007                         301       CL_WHT  EQU     CL_GRN+CL_RED+CL_BLU    ; White
                               302 +1    $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           317     ;;      MTR_DFC - Display Font Character
                           318     ;
                           319     ;       MTR_DFC displays the indexed font character at the specified
                           320     ;       screen address.  The value of compf is XOR'd with each of
                           321     ;       the bytes displayed, thus, the character will be displayed
                           322     ;       straight if compf is 0, or it will be displayed in reverse
                           323     ;       video if compf is 0FFFFH.
                           324     ;
                           325     ;       Usage:  CALL MTR_DFC(horz,vert,findx,compf)
                           326     ;
                           327     ;       Entry:
                           328     ;           horz        = Horizontal character index within line [0,79]  BYTE
                           329     ;           vert        = Vertical character index within screen [0,24]  BYTE
                           330     ;           findx       = Index into Font ROM for the character  [0,129] BYTE
                           331     ;           compf       = Mask to be XOR'd with display bytes            WORD
                           332     ;       Globals:
                           333     ;           Color structure initialized
                           334     ;
                           335     ;       Exit:   NONE
                           336     ;
                           337
                           338
.....                      339     P_DFC           STRUC
0000                       340                     DW      ?
0002                       341                     DW      ?
0004                       342                     DW      ?
0006                       343      DFC_COMPF      DW      ?       ; A real WORD
0008                       344      DFC_FINDX      DB      ?       ; Font Character Index
0009                       345                     DB      ?
000A                       346      DFC_VERT       DB      ?       ; Vertical Character Line
000B                       347                     DB      ?
000C                       348      DFC_HORZ       DB      ?       ; Horizontal Character Column
000D                       349                     DB      ?
.....                      350       P_DFC ENDS
                           351
                           352
0001                       353     MTR_DFC PROC    FAR
                           354             PUBLIC  MTR_DFC
0001 55                    355             PUSH    BP
0002 8BEC                  356             MOV     BP,SP
                           357
0004 1E                    358             PUSH    DS
0005 E4D8                  359             IN      AL,VID_CMD
0007 50                    360             PUSH    AX                      ; Save current parameters
                           361
                           362     ;       Compute offset of character in Video RAM
                           363
0008 8A5E0C                364             MOV     BL,[BP].DFC_HORZ        ; BL = Horizontal Character Index
000B 8A7E0A                365             MOV     BH,[BP].DFC_VERT        ; BH = Vertical Line Index
000E E80603                366             CALL    CCA                     ; Compute Character Address
                           367
                           368     ;       Compute offset of character in Font ROM
                           369
0011 8A4E08                370             MOV     CL,[BP].DFC_FINDX       ; CL = Font ROM Character Index
0014 B500                  371             MOV     CH,0
```

```
LOC  OBJ                  LINE     SOURCE

                          427
                          428              POP     BX              ; BL = COLOR.PFONT
006B E8B602               429              CALL    SWP             ; Set Write Parameters
006E 7403                 430              JZ      DFC4            ; No Foreground Color
0070 E83700               431              CALL    DFC5            ; Display the character
0073                      432      DFC4:   LABEL   NEAR
0073 B307                 433              MOV     BL,CL_WHT
0075 E8AC02               434              CALL    SWP             ; Write to all planes of VRAM
0078 EB13                 435              JMP     SHORT DFC4_6
                          436
007A A00200          E    437      DFC4_3: MOV     AL,COLOR.MSK
007D 50                   438              PUSH    AX              ; Save Color Mask
007E 8B4E06               439              MOV     CX,[BP].DFC_COMPF ; CX = Complement Flag
0081 C5060000        E    440              LDS     AX,FONT
0085 03F0                 441              ADD     SI,AX           ; DS:SI = Pointer to Font Character
0087 E8D602               442              CALL    SWPG            ; Set Write Parameters to all Green
008A E81D00               443              CALL    DFC5
008D                      444      DFC4_6: LABEL   NEAR
                          445
                          446      ;       Save Character Attributes
                          447
                          448
008D 8A4608               449              MOV     AL,[BP].DFC_FINDX       ; Save font Index
0090 2688858004           450              MOV     ES:BYTE PTR [DI+9*128],AL
                          451
0095 58                   452              POP     AX              ; AL = COLOR.MSK
0096 243F                 453              AND     AL,00111111B    ; Set Color Mask
0098 80E580               454              AND     CH,080H
009B 0AC5                 455              OR      AL,CH           ; Set Reverse Video Flag
009D 2688850005           456              MOV     ES:BYTE PTR [DI+10*128],AL      ; Save Character Attributes
                          457
00A2 58                   458              POP     AX
00A3 E6D8                 459              OUT     VID_CMD,AL      ; Restore original video register
00A5 1F                   460              POP     DS
                          461
00A6 5D                   462              POP     BP
00A7 CA0800               463              RET     8
                          464
                          465      MTR_DFC ENDP
                          466
                          467
                          468
00AA                      469      DFC5    PROC    NEAR
00AA 56                   470              PUSH    SI
                          471
00AB AD                   472              LODSW                   ; AX = Font character (0,1)
00AC 33C1                 473              XOR     AX,CX           ; Complent bits for reverse video
00AE 268805               474              MOV     ES:BYTE PTR [DI+0*128],AL
00B1 2688A58000           475              MOV     ES:BYTE PTR [DI+1*128],AH
00B6 AD                   476              LODSW                   ; AX = Font character (2,3)
00B7 33C1                 477              XOR     AX,CX
00B9 2688850001           478              MOV     ES:BYTE PTR [DI+2*128],AL
00BE 2688A58001           479              MOV     ES:BYTE PTR [DI+3*128],AH
00C3 AD                   480              LODSW                   ; AX = Font character (4,5)
00C4 33C1                 481              XOR     AX,CX
```

```
LOC  OBJ                  LINE     SOURCE

                           519     ;;      MTR_EDC - Erase Display Character
                           520     ;
                           521     ;       MTR_EDC erases the specified character from the screen.
                           522     ;       In this case, it mearly zeroes out video RAM.
                           523     ;
                           524     ;       NOTE:   This routine assumes that all of the char-
                           525     ;               acters are on the same line!  If they are
                           526     ;               not, bad video RAM may be smashed due to
                           527     ;               the memory mapping scheme (in a 32KB system).
                           528     ;
                           529     ;       Usage:          CALL MTR_EDC(line,char,count);
                           530     ;
                           531     ;       Parameters:
                           532     ;
                           533     ;           line        - Line of character to erase, BYTE, [0-24]
                           534     ;           char        - First Character to erase, BYTE, [0-79]
                           535     ;           count       - Number of characters to erase, BYTE, [0-79]
                           536     ;
                           537     ;       Exit:   NONE
                           538     ;
                           539
------                     540     P_EDC   STRUC
0000                       541                     DW      ?
0002                       542                     DW      ?
0004                       543                     DW      ?
0006                       544      EDC_COUNT      DB      ?       ; Count
0007                       545                     DB      ?
0008                       546      EDC_CHAR       DB      ?       ; Character
0009                       547                     DB      ?
000A                       548      EDC_LINE       DB      ?       ; Line
000B                       549                     DB      ?
------                     550       P_EDC         ENDS
                           551
                           552
010E                       553     MTR_EDC PROC    FAR
                           554             PUBLIC  MTR_EDC
010E 55                    555             PUSH    BP
010F 8BEC                  556             MOV     BP,SP
                           557
0111 E4D8                  558             IN      AL,VID_CMD
0113 50                    559             PUSH    AX              ; Save Video Parameters
                           560
                           561     ;       Fetch Parameters
                           562
0114 8A7E0A                563             MOV     BH,[BP].EDC_LINE; BH = Row
0117 8A5E08                564             MOV     BL,[BP].EDC_CHAR; BL = Column
011A E80102                565             CALL    CCA_            ; DI = Offset for First Character
011D 8A5606                566             MOV     DL,[BP].EDC_COUNT
0120 B600                  567             MOV     DH,0            ; DX = Character Count
0122 B90900                568             MOV     CX,SLPC-2       ; CX = Scan Lines per Character
                           569
                           570     ;       Zero the characters one scan line at a time
                           571
0125 FC                    572             CLD                     ; Auto-Increment
0126 51                    573     EDC1:   PUSH    CX              ; Save Scan Line Count
```

```
LOC  OBJ                  LINE      SOURCE

                           628 +1   $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

018A 8BF3                  684              MOV     SI,BX           ; SI = Source Character Address
018C 8A5606                685              MOV     DL,[BP].MDC_COUNT
018F B600                  686              MOV     DH,0            ; DX = Move Count
0191 B90B00                687              MOV     CX,SLPC         ; CX = Scan Lines per Character
                           688
0194 83FA00                689              CMP     DX,0
0197 744E                  690              JE      MDC4            ; No characters to move
0199 FC                    691              CLD                     ; Assume Auto-Increment
019A B80000                692              MOV     AX,0            ; Assume no negative word adjust
019D 3BFE                  693              CMP     DI,SI
019F 7213                  694              JB      MDC1
                           695
                           696      ;       Destination >= Source
                           697
01A1 FD                    698              STD                     ; Auto-Decrement
01A2 B8FFFF                699              MOV     AX,-1           ; Back each address up one byte before movsw
01A5 03FA                  700              ADD     DI,DX
01A7 4F                    701              DEC     DI
01A8 03F2                  702              ADD     SI,DX
01AA 4E                    703              DEC     SI              ; Advance Pointers to the end of strings
                           704
01AB 8A1E0500        E     705              MOV     BL,H19_MODE.BWO
01AF 80FBFF                706              CMP     BL,PLM_TRUE
01B2 741D                  707              JZ      MDC2            ; Optimize Black & White
                           708
                           709      ;       Move Color Characters
                           710
01B4 51                    711      MDC1:   PUSH    CX
01B5 B304                  712              MOV     BL,CL_GRN
01B7 E83500                713              CALL    MDC5            ; Move Green Plane
01BA B302                  714              MOV     BL,CL_RED
01BC E83000                715              CALL    MDC5            ; Move Red   Plane
01BF B301                  716              MOV     BL,CL_BLU
01C1 E82B00                717              CALL    MDC5            ; Move Blue  Plane
01C4 59                    718              POP     CX
                           719
01C5 81C78000              720              ADD     DI,128
01C9 81C68000              721              ADD     SI,128
01CD E2E5                  722              LOOP    MDC1
01CF EB16                  723              JMP     SHORT MDC4
                           724
                           725      ;       Optimized Black & White Character Move
                           726
01D1 50                    727      MDC2:   PUSH    AX
01D2 E88B01                728              CALL    SWPG            ; Set Multiple Write for all planes
01D5 8ED8                  729              MOV     DS,AX           ; Source Plane = Destination Plane
01D7 58                    730              POP     AX
                           731
01D8 51                    732      MDC3:   PUSH    CX
01D9 E81A00                733              CALL    MDC6
01DC 59                    734              POP     CX
01DD 81C78000              735              ADD     DI,128
01E1 81C68000              736              ADD     SI,128
01E5 E2F1                  737              LOOP    MDC3
                           738
```

```
LOC  OBJ                  LINE     SOURCE

                          776
                          777
                          778
                          779
                          780      ;;       MTR_MDL - Move Display Line
                          781      ;
                          782      ;        MTR_MDL moves the line of characters to the spec-
                          783      ;        ified destination line.
                          784      ;
                          785      ;        Usage:          CALL MTR_MDL(line_src,line_dst);
                          786      ;
                          787      ;        Parameters:
                          788      ;
                          789      ;             line_src    - Source Line
                          790      ;             line_dst    - Destination Line
                          791      ;
                          792      ;        Exit:   NONE
                          793      ;
                          794
-----                     795      P_MDL    STRUC
0000                      796                        DW      ?
0002                      797                        DW      ?
0004                      798                        DW      ?
0006                      799       MDL_DST          DB      ?               ; Destination Line
0007                      800                        DB      ?
0008                      801       MDL_SRC          DB      ?               ; Source Line
0009                      802                        DB      ?
-----                     803        P_MDL ENDS
                          804
                          805
0208                      806      MTR_MDL  PROC     FAR
                          807               PUBLIC   MTR_MDL
0208 55                   808               PUSH     BP
0209 8BEC                 809               MOV      BP,SP
020B 1E                   810               PUSH     DS
020C E4D8                 811               IN       AL,VID_CMD
020E 50                   812               PUSH     AX
                          813
                          814      ;        Initialize from Parameters
                          815
020F 8A7E06               816               MOV      BH,[BP].MDL_DST
0212 B300                 817               MOV      BL,0
0214 E80701               818               CALL     CCA_            ; DI,BX = Destination Line/Char Offset
0217 8A7E08               819               MOV      BH,[BP].MDL_SRC
021A E8FA00               820               CALL     CCA             ; BX = Source Line Offset
021D 8BF3                 821               MOV      SI,BX           ; SI = Source Character Index
                          822
021F B600                 823               MOV      DH,0
0221 8A160000         E   824               MOV      DL,H19_PAR.CPL  ; DX = Number of Characters Per Line
0225 B90B00               825               MOV      CX,SLPC         ; CX = Scan Lines per Character
0228 FC                   826               CLD                      ; Auto Increment
                          827
0229 8A1E0500         E   828               MOV      BL,H19_MODE.BWO
022D 80FBFF               829               CMP      BL,PLM_TRUE
0230 741D                 830               JE       MDL2            ; Optimize this line move
```

```
LOC  OBJ                  LINE      SOURCE

027D C3                    885              RET
                           886
                           887      MDL5    ENDP
                           888
                           889
                           890
                           891
                           892 +1   $EJECT
```

```
LOC  OBJ                  LINE      SOURCE

02BE 00
02BF 00
02C0 00
02C1 C0                    948                     DB      11000000B,00111111B,11111111B,11111100B
02C2 3F
02C3 FF
02C4 FC
02C5 7C                    949                     DB      01111100B,11000000B,01000000B,00000010B
02C6 C0
02C7 40
02C8 02
02C9 47                    950                     DB      01000111B,00000000B,00111111B,11111100B
02CA 00
02CB 3F
02CC FC
02CD 44                    951                     DB      01000100B,00001110B,00100010B,00000000B
02CE 0E
02CF 22
02D0 00
02D1 47                    952                     DB      01000111B,00000001B,11111100B,00000000B
02D2 01
02D3 FC
02D4 00
02D5 7C                    953                     DB      01111100B,11000001B,00000100B,00000000B
02D6 C1
02D7 04
02D8 00
02D9 C0                    954                     DB      11000000B,00111111B,11111000B,00000000B
02DA 3F
02DB F8
02DC 00
02DD 00                    955                     DB      00000000B,00000000B,00000000B,00000000B
02DE 00
02DF 00
02E0 00
                           956
                           957      MTR_PROMPT     ENDP
                           958
                           959
                           960
                           961
                           962 +1   $EJECT
```

```
LOC  OBJ                  LINE      SOURCE

                          1018      MTR_RDC ENDP
                          1019
                          1020
                          1021
                          1022
                          1023 +1   $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                          1065
                          1066
                          1067
                          1068
                          1069     ;;       CCA      - Compute Character Address
                          1070     ;
                          1071     ;        CCA returns the character offset in BX based
                          1072     ;        on the character Row and Column.
                          1073     ;
                          1074     ;        Entry:  BH       - Character Row
                          1075     ;                BL       - Character Column
                          1076     ;
                          1077     ;        Exit:   BX       - Character Offset within Video Plane
                          1078     ;
                          1079     ;        Uses:   BH
                          1080     ;
                          1081
0317                      1082     CCA      PROC    NEAR
0317 D0E7                 1083              SHL     BH,1
0319 D0E7                 1084              SHL     BH,1
031B D0E7                 1085              SHL     BH,1
031D C3                   1086              RET
                          1087     CCA      ENDP
                          1088
                          1089
031E                      1090     CCA_     PROC    NEAR
031E E8F6FF               1091              CALL    CCA
0321 8BFB                 1092              MOV     DI,BX
0323 C3                   1093              RET
                          1094     CCA_     ENDP
                          1095 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

0350 0000                 1151     SWPB    DW      00000H          ; Black
0352 00C0                 1152             DW      PAR_BLU         ; Blue
0354 00D0                 1153             DW      PAR_RED         ; Red
0356 00D0                 1154             DW      PAR_RED         ; Magenta
0358 00E0                 1155             DW      PAR_GRN         ; Green
035A 00E0                 1156             DW      PAR_GRN         ; Cyan
035C 00E0                 1157             DW      PAR_GRN         ; Yellow
035E 00E0                 1158             DW      PAR_GRN         ; White
                          1159
                          1160     SWP     ENDP
                          1161
                          1162
                          1163
                          1164
0360                      1165     SWPG    PROC    NEAR
0360 E4D8                 1166             IN      AL,VID_CMD
0362 240F                 1167             AND     AL,00001111B
0364 E6D8                 1168             OUT     VID_CMD,AL
0366 B800E0               1169             MOV     AX,PAR_GRN
0369 23C0                 1170             AND     AX,AX           ; FLAGS = NZ
036B 8EC0                 1171             MOV     ES,AX
036D C3                   1172             RET
                          1173
                          1174     SWPG    ENDP
                          1175
                          1176
                          1177
                          1178
                          1179     VIDEOA  ENDS
                          1180
                          1181             END
```

```
NAME            TYPE     VALUE  ATTRIBUTES, XREFS

DCI . . . . . V DWORD  0000H  EXTRN  235#
DFC . . . . . V DWORD  0000H  EXTRN  236#
DFC_COMPF . . . V WORD   0006H  S FIELD  343# 413 439
DFC_FINDX . . . V BYTE   0008H  S FIELD  344# 370 449
DFC_HORZ. . . . V BYTE   000CH  S FIELD  348# 364
DFC_VERT. . . . V BYTE   000AH  S FIELD  346# 365
DFC1. . . . . L NEAR   003AH  VIDEOA  392 395#
DFC2. . . . . L NEAR   0048H  VIDEOA  401 404#
DFC3. . . . . L NEAR   006AH  VIDEOA  420 424#
DFC4. . . . . L NEAR   0073H  VIDEOA  430 432#
DFC4_3. . . . . L NEAR   007AH  VIDEOA  386 437#
DFC4_6. . . . . L NEAR   008DH  VIDEOA  435 444#
DFC5. . . . . L NEAR   00AAH  VIDEOA  422 431 443 469# 495
DFC6. . . . . L NEAR   00E7H  VIDEOA  394 403 500# 517
DS_SIZE . . . . NUMBER   0040H    63#
DSC . . . . . V BYTE   0001H  S FIELD  146#
EDC . . . . . V DWORD  0000H  EXTRN  238#
EDC_CHAR. . . . V BYTE   0008H  S FIELD  546# 564
EDC_COUNT . . . V BYTE   0006H  S FIELD  544# 566
EDC_LINE. . . . V BYTE   000AH  S FIELD  548# 563
EDC1. . . . . L NEAR   0126H  VIDEOA  573# 593
EDC2. . . . . L NEAR   013AH  VIDEOA  579 582#
EDC3. . . . . L NEAR   0146H  VIDEOA  586 589#
EDC4. . . . . L NEAR   016BH  VIDEOA  581 588 600 604 616# 627
EDC5. . . . . L NEAR   0173H  VIDEOA  621 623#
EMEC. . . . . V DWORD  0000H  EXTRN  239#
ESCP. . . . . V TBYTE  0000H  EXTRN  253# 1038
ESCP_STRUC. . . STRUC          SIZE=000AH  #FIELDS=6  115 122# 253
EXPAND. . . . . V BYTE   0008H  S FIELD  133#
FONT. . . . . V DWORD  0000H  EXTRN  240# 414 440
FONTAB. . . . . SEGMENT        SIZE=0000H BYTE  PUBLIC 'DATA'  180# 182
FORE. . . . . V BYTE   0000H  S FIELD  101#
FUNCTION. . . . V WORD   0001H  S FIELD  117#
GRAPHIC . . . . V BYTE   0009H  S FIELD  134#
H19_MODE. . . . V   18  0000H  EXTRN  254# 384 705 828 1037
H19_MODE_STRUC. STRUC          SIZE=0012H  #FIELDS=17  124 142# 254
H19_PAR . . . . V     9  0000H  EXTRN  255# 824
H19_PAR_STRUC . STRUC          SIZE=0009H  #FIELDS=9  144 154# 255
HORZ_CHAR . . . V BYTE   0000H  EXTRN  230# 911
INDEX . . . . . V BYTE   0000H  S FIELD  94#
INIT. . . . . L NEAR   0000H  EXTRN  222#
INSERT. . . . . V BYTE   000AH  S FIELD  135#
ITV_OBI . . . . V DWORD  0000H  EXTRN  189#
ITV_SST . . . . V DWORD  0000H  EXTRN  188#
KEY_EN. . . . . V BYTE   000BH  S FIELD  136#
LFS . . . . . V BYTE   0002H  S FIELD  147#
MDC . . . . . V DWORD  0000H  EXTRN  241#
MDC_COUNT . . . V BYTE   0006H  S FIELD  659# 685
MDC_DST . . . . V BYTE   0003H  S FIELD  661# 681
MDC_LINE. . . . V BYTE   000CH  S FIELD  665# 680
MDC_SRC . . . . V BYTE   000AH  S FIELD  663# 683
MDC1. . . . . L NEAR   01B4H  VIDEOA  694 711# 722
MDC2. . . . . L NEAR   01D1H  VIDEOA  707 727#
MDC3. . . . . L NEAR   01D8H  VIDEOA  732# 737
MDC4. . . . . L NEAR   01E7H  VIDEOA  690 723 739#
```

```
NAME            TYPE     VALUE  ATTRIBUTES, XREFS

PAR_VID . . . . NUMBER   C000H    67#
PFONT . . . . . V BYTE   0005H  S FIELD  106# 410 928
PLM_TRUE. . . . NUMBER   00FFH    82# 385 706 829
PORT. . . . . V BYTE   0001H  S FIELD  95#
POVRAM. . . . . V BYTE   0003H  S FIELD  148#
PROMPT. . . . . V DWORD  0000H  EXTRN  243#
RDC . . . . . V DWORD  0000H  EXTRN  244#
RDC_HORZ. . . . V BYTE   0006H  S FIELD  989# 1006
RDC_VERT. . . . V BYTE   0008H  S FIELD  991# 1007
RDC_XC_PTR. . . V DWORD  000AH  S FIELD  993# 1011
REGF. . . . . V DWORD  0000H  EXTRN  232#
RESETF. . . . . V BYTE   0000H  EXTRN  233#
REVERSE . . . . V WORD   000CH  S FIELD  137#
ROW . . . . . V BYTE   0006H  S FIELD  161#
S_CRT . . . . . L NEAR   0000H  EXTRN  215#
S_KBD . . . . . L NEAR   0000H  EXTRN  216#
S_XMTC. . . . . V DWORD  0000H  EXTRN  245#
SDS . . . . . L NEAR   0000H  EXTRN  204#
SHIFTED . . . . V BYTE   000EH  S FIELD  138#
SLI . . . . . V BYTE   0004H  S FIELD  149#
SLPC. . . . . NUMBER   000BH    275# 568 687 825 930
SPC . . . . . V BYTE   0005H  S FIELD  150#
SS_SIZE . . . . NUMBER   0020H    62#
START . . . . . V WORD   0000H  S FIELD  111#
STATUS. . . . . V BYTE   000FH  S FIELD  139#
STRNG . . . . . V BYTE   0002H  S FIELD  96#
SW401 . . . . . V BYTE   0006H  S FIELD  151#
SW402 . . . . . V BYTE   0007H  S FIELD  152#
SWP . . . . . L NEAR   0324H  VIDEOA  391 400 419 429 434 578 585 598 752 872 929 1119# 1160
SWP1. . . . . L NEAR   032AH  VIDEOA  1121 1124#
SWPA. . . . . V BYTE   0348H  VIDEOA  1128 1140#
SWPB. . . . . V WORD   0350H  VIDEOA  1133 1151#
SWPG. . . . . L NEAR   0360H  VIDEOA  442 728 850 1004 1165# 1174
TTY_INTR. . . . L NEAR   0000H  EXTRN  218#
TTY_POLL. . . . L NEAR   0000H  EXTRN  219#
TXMT. . . . . L NEAR   0000H  EXTRN  217# 1056
UIES. . . . . V DWORD  0000H  EXTRN  246#
UNIT. . . . . V BYTE   0052H  S FIELD  97#
UPDATE. . . . . V BYTE   0002H  S FIELD  112#
UPDN. . . . . V BYTE   0010H  S FIELD  140#
VERT_LINE . . . V BYTE   0000H  EXTRN  231# 910
VID_CMD . . . . NUMBER   00D8H    277# 359 459 558 607 675 741 811 862 905 941 1001 1015 1126 1131 1166 1168
VIDEO_INTR. . . L NEAR   0000H  EXTRN  212#
VIDEO_INTR_A. . L FAR    0000H  EXTRN  174#
VIDEOA. . . . . SEGMENT        SIZE=036EH BYTE  PUBLIC 'CODE'  264# 271 1179
VRAM_SIZE . . . V BYTE   0008H  S FIELD  153#
WRAP. . . . . V BYTE   0011H  S FIELD  141#
WRITEC. . . . . L NEAR   0000H  EXTRN  223# 921
XCA . . . . . V DWORD  0000H  EXTRN  247#
XMT . . . . . V TBYTE  0000H  EXTRN  256#
XMT_COLOR . . . V BYTE   0007H  S FIELD  162#
XMT_GRAPHIC . . V BYTE   0008H  S FIELD  163#
XMT_REVERSE . . V BYTE   0009H  S FIELD  164#
XMT_STRUC . . . STRUC          SIZE=000AH  #FIELDS=8  156 165# 256
```

```
SERIES-III 8086/8087/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE B207A
OBJECT MODULE PLACED IN :F2:B207A.OBJ
INVOCATION LINE CONTROLS:  PRINT(:TO:) XREF ERRORPRINT(:TO:)

LOC  OBJ                  LINE     SOURCE
```

```
LOC  OBJ                  LINE     SOURCE

0002                   =2    50       MSK             DB      ?       ; SHL(Back,3) OR Fore
0003                   =2    51       CLEAR           DB      ?       ; Color to Clear
0004                   =2    52       PAINTED         DB      ?       ; Color to Fully Paint
0005                   =2    53       PFONT           DB      ?       ; Color to set to Font Pattern
0006                   =2    54       COMP_FONT       DB      ?       ; Color to set to Complement of Font Pattern
------                 =2    55         COLOR_STRUC   ENDS
                       =2    56
------                 =2    57     CRTC_STRUC        STRUC
0000                   =2    58         START         DW      ?       ; CRT-C ..... Start Address
0002                   =2    59         UPDATE        DB      ?       ; True if update is required
------                 =2    60       CRTC_STRUC      ENDS
                       =2    61
------                 =2    62     ESCP_STRUC        STRUC
0000                   =2    63         COMMAND       DB      ?       ; Current Escape Command
0001                   =2    64         FUNCTION      DW      ?       ; Pointer to Escape Processor
0003                   =2    65         MODE          DB      ?       ; Escape Emulation Mode
0004                   =2    66         OPER_COUNT    DB      ?       ; Operand Count
0005                   =2    67         OPER_INDEX    DB      ?       ; Operand Index
0006                   =2    68         OPERAND       DB      4 DUP(?); Operands
------                 =2    69       ESCP_STRUC      ENDS
                       =2    70
------                 =2    71     H19_MODE_STRUC    STRUC
0000                   =2    72         ALTERNATE     DB      ?       ; Alternate Key-pad Mode
0001                   =2    73         ANSI          DB      ?       ; ANSI Mode
0002                   =2    74         AUTO_CR       DB      ?       ; Auto Carriage-Return on Line-Feed
0003                   =2    75         AUTO_LF       DB      ?       ; Auto Line-Feed on Carriage-Return
0004                   =2    76         AUTO_REPEAT   DB      ?       ; Auto-Repeat Keyboard
0005                   =2    77         BWO           DB      ?       ; Black and White Optimization
0006                   =2    78         CURSOR        DB      ?       ; Programmed Cursor Value
0007                   =2    79         CURSOR_ON     DB      ?       ; Cursor Enabled
0008                   =2    80         EXPAND        DB      ?       ; Expand Key-Board Characters
0009                   =2    81         GRAPHIC       DB      ?       ; Graphic Character Mode
000A                   =2    82         INSERT        DB      ?       ; Insert Character Mode
000B                   =2    83         KEY_EN        DB      ?       ; Key-Board Enable
000C                   =2    84         REVERSE       DW      ?       ; Reverse Video
000E                   =2    85         SHIFTED       DB      ?       ; Shifted Key-Pad Mode
000F                   =2    86         STATUS        DB      ?       ; Status-Line Enabled
0010                   =2    87         UPDN          DB      ?       ; Key-Board Up/Down Mode
0011                   =2    88         WRAP          DB      ?       ; Wrap at End-of-Line
------                 =2    89       H19_MODE_STRUC          ENDS
                       =2    90
------                 =2    91     H19_PAR_STRUC     STRUC
0000                   =2    92         CPL           DB      ?       ; Characters Per Line
0001                   =2    93         DSC           DB      ?       ; Displayed Scan Lines per Character
0002                   =2    94         LPS           DB      ?       ; Lines Per Screen
0003                   =2    95         POVRAM        DB      ?       ; Planes of Video RAM
0004                   =2    96         SLI           DB      ?       ; Status Line Index
0005                   =2    97         SPC           DB      ?       ; Scan Lines per Character
0006                   =2    98         SW401         DB      ?       ; H19-Switch 401
0007                   =2    99         SW402         DB      ?       ; H19-Switch 402
0008                   =2   100         VRAM_SIZE     DB      ?       ; 0-32KBytes, 1-64KBytes
------                 =2   101       H19_PAR_STRUC   ENDS
                       =2   102
------                 =2   103     XMT_STRUC         STRUC
0000                   =2   104         BURST         DB      ?       ; Characters to Transmit per VSYNC
```

```
LOC  OBJ                  LINE     SOURCE

                    =1                          EXTRN   MTR101:NEAR      ; PL/M-86 Monitor Loop
                    =1                          EXTRN   VIDEO_INTR:NEAR ; PL/M-86 Video Interrupt
                    =1                          EXTRN   D_CRT:NEAR      ; PL/M-86 Dumb Terminal
                    =1                          EXTRN   D_KBD:NEAR      ; PL/M-86 Dumb Key-Board
                    =1                          EXTRN   S_CRT:NEAR      ; PL/M-86 Smart Terminal
                    =1                          EXTRN   S_KBD:NEAR      ; PL/M-86 Smart Key-Board
                    =1                          EXTRN   TXMT:NEAR       ; PL/M-86 Transmit Screen Character
                    =1                          EXTRN   TTY_INTR:NEAR   ; PL/M-86 Terminal Interrupt
                    =1                          EXTRN   TTY_POLL:NEAR   ; PL/M-86 Interrupt Poll
                    =1
                    =1                          EXTRN   CRLF:NEAR       ; PL/M-86 CR-LF
                    =1                          EXTRN   INIT:NEAR       ; PL/M-86 Initialize Hardware
                    =1                          EXTRN   WRITEC:NEAR     ; PL/M-86 Write Character
                    =1                  CODE    ENDS
                    =1                  )FI
------              =1      169
                    =1      170
                    =1      171         %IF(%NES(DATA,%SEGMENT_LABEL)) THEN (
                    =1                    DATA  SEGMENT WORD PUBLIC 'DATA'
                    =1                          EXTRN   HORZ_CHAR:BYTE  ; Column Index
                    =1                          EXTRN   VERT_LINE:BYTE  ; Row    Index
                    =1                          EXTRN   REGP:DWORD      ; Pointer to Saved Processor Registers
                    =1                          EXTRN   RESETF:BYTE     ; Hardware Reset Flag
                    =1
                    =1                          EXTRN   DCI:DWORD       ; Display Character Initialization
                    =1                          EXTRN   DFC:DWORD       ; Display Font Character
                    =1                          EXTRN   D_XMTC:DWORD    ; Dumb Terminal Transmit Character
                    =1                          EXTRN   EDC:DWORD       ; Erase Display Character
                    =1                          EXTRN   EMEC:DWORD      ; Extend-Mode Escape Character
                    =1                          EXTRN   FONT:DWORD      ; Pointer to Font Table
                    =1                          EXTRN   MDC:DWORD       ; Move Display Character
                    =1                          EXTRN   MDL:DWORD       ; Move Display Line
                    =1                          EXTRN   PROMPT:DWORD    ; Display Monitor Prompt
                    =1                          EXTRN   RDC:DWORD       ; Read Display Character
                    =1                          EXTRN   S_XMTC:DWORD    ; Smart Terminal Transmit Character
                    =1                          EXTRN   UIES:DWORD      ; Un-Implemented Escape Sequence
                    =1                          EXTRN   XCA:DWORD       ; Transmit Character Attributes
                    =1
                    =1                          EXTRN   BOOT_PAR:BOOT_PAR_STRUC
                    =1                          EXTRN   COLOR:COLOR_STRUC
                    =1                          EXTRN   CRTC_CURSOR:CRTC_STRUC
                    =1                          EXTRN   CRTC_DISPLAY:CRTC_STRUC
                    =1                          EXTRN   ESCP:ESCP_STRUC
                    =1                          EXTRN   H19_MODE:H19_MODE_STRUC
                    =1                          EXTRN   H19_PAR:H19_PAR_STRUC
                    =1                          EXTRN   XMT:XMT_STRUC
                    =1                    DATA  ENDS
                    =1                    )FI
------              =1      202
                    =1      203
                    =1      204         %IF(%NES(VIDEOA,%SEGMENT_LABEL)) THEN (
                    =1                    VIDEOA  SEGMENT BYTE PUBLIC 'CODE'
                    =1                    VIDEOA  ENDS
                    =1                    )FI
------              =1      208
```

```
LOC  OBJ                  LINE     SOURCE

                          242      ;;      CSP     - Controller Status Port
                          243      ;
                          244      ;       CSP returns the value for the controller status port
                          245      ;
                          246      ;       Entry:  NONE
                          247      ;
                          248      ;       Exit:   AL      = Controller Status Port Value
                          249      ;
                          250      ;       Uses:   AL,DH
                          251      ;
                          252
0000                      253      CSP     PROC    NEAR
                          254              PUBLIC  CSP
                          255
0000 8A160100       E     256              MOV     DL,BOOT_PAR.PORT
0004 B600                 257              MOV     DH,0                ; DX = Device Port
0006 EC                   258              IN      AL,DX
0007 C3                   259              RET
                          260
                          261      CSP     ENDP
                          262 +1   $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           300     ;;      RDTRACKA        - Read Track
                           301     ;
                           302     ;       RDTRACKA is the assembly language assist routine
                           303     ;       for RDTRACK.  This routine is required to perform
                           304     ;       the actual disk transfer, as the code may be opt-
                           305     ;       timized.
                           306     ;
                           307     ;       Usage:  CALL RDTRACKA(base_port,cmd,cnt,ptr$data)
                           308     ;
                           309     ;       Entry:  base_port       = Base Port Address for device
                           310     ;               cmd             = command
                           311     ;               cnt             = Byte Count
                           312     ;               ptr$data        = Address to save data at
                           313     ;
                           314     ;       Exit:   AX      = Byte Count
                           315     ;
                           316     ;       Uses:   ???
                           317     ;
                           318
                           319
                           320     RDTA    STRUC
0000                       321                     DW      ?
0002                       322                     DW      ?
0004                       323      PTR_DATA       DD      ?       ; Address to save data at
0008                       324      CNT            DW      ?       ; Byte Count
000A                       325      CMD            DW      ?       ; Command
000C                       326      BASE_PORT      DW      ?       ; Base Port Address for device
                           327      RDTA    ENDS
                           328
                           329
0019                       330     RDTRACKA        PROC    NEAR
                           331             PUBLIC  RDTRACKA
                           332
0019 55                    333             PUSH    BP
001A 8BEC                  334             MOV     BP,SP
                           335
001C 8B560C                336             MOV     DX,[BP].BASE_PORT
001F 8B460A                337             MOV     AX,[BP].CMD             ; AX = Command
0022 EE                    338             OUT     DX,AL                   ; Output the command to command port
                           339
0023 FC                    340             CLD                             ; Set auto-increment
0024 33DB                  341             XOR     BX,BX                   ; BX = 0
0026 8B4E08                342             MOV     CX,[BP].CNT             ; CX = Byte Count
0029 C47E04                343             LES     DI,[BP].PTR_DATA        ; ES:DI = Address to store data at
002C 83C203                344             ADD     DX,FD_DATA              ; DX = Data Port Address
002F EC                    345     RDT1:   IN      AL,DX                   ; Input Data Byte
0030 AA                    346             STOSB
0031 43                    347             INC     BX
0032 E2FB                  348             LOOP    RDT1                    ; Read another sector
                           349
0034 8BC3                  350             MOV     AX,BX                   ; AX = Actual read count
0036 5D                    351             POP     BP
0037 C20A00                352             RET     10                      ; Discard the parameters
                           353
                           354     RDTRACKA        ENDP
```

```
LOC  OBJ                  LINE     SOURCE

                          360
                          361
                          362
                          363
                          364      ;;       WDR      - Wait for Drive Ready
                          365      ;
                          366      ;        WDR waits until the selected drive is ready by waiting
                          367      ;        for at least 7 index hole transitions.
                          368      ;
                          369      ;        Entry:   NONE
                          370      ;
                          371      ;        Exit:    TIME_OUT = results of loop time-out.
                          372      ;
                          373      ;        Uses:    NONE
                          374      ;
                          375
003A                      376      WDR      PROC     NEAR
                          377               PUBLIC   WDR
                          378
003A BB581B90             379               MOV      BX,WDRA          ; BX = Time-Out Loop counter
003E B9041E               380               MOV      CX,4+30*256      ; CL = Number of Index Hole transitions
                          381                                         ; CH = Number of Seconds for Time-Out
                          382
0041 E81600               383      WDR1:    CALL     WDR4             ; Wait while hole
0044 720D                 384               JC       WDR3             ; Operation Timed Out
0046 75F9                 385               JNZ      WDR1             ; Still hole
                          386
0048 E80F00               387      WDR2:    CALL     WDR4             ; Wait while hole
004B 7206                 388               JC       WDR3             ; Operation Timed out
004D 74F9                 389               JZ       WDR2             ; Still no hole
                          390
004F FEC9                 391               DEC      CL
0051 75EE                 392               JNZ      WDR1             ; Looking for more index holes
                          393
0053 0ADD                 394      WDR3:    OR       BL,CH            ; Insure BX <> 0 if no time-out occured
0055 891E0000         E   395               MOV      TIME_OUT,BX      ; Save time-out results
0059 C3                   396               RET
                          397
  1B58                    398      WDRA     EQU      7000             ; Approx. 1 second for time-out loop
                          399
                          400      WDR      ENDP
                          401
                          402
005A                      403      WDR4     PROC     NEAR
                          404
005A 4B                   405               DEC      BX               ; Decrement Time-Out Counter
005B 7507                 406               JNZ      WDR5             ; Device did NOT Time Out
005D FECD                 407               DEC      CH
005F 7414                 408               JZ       WDR6             ; Device DID Time Out
0061 BB581B               409               MOV      BX,WDRA          ; Re-Initialize Low-Order Loop Counter
                          410
0064 53                   411      WDR5:    PUSH     BX
0065 51                   412               PUSH     CX
0066 E80000           E   413               CALL     CBA              ; Check for Boot Abort
0069 59                   414               POP      CX
```

```
LOC  OBJ                  LINE     SOURCE

                          432      ;;      WNB     - Wait for NOT Busy
                          433      ;
                          434      ;       WNB waits for the controller to deassert busy.  This is
                          435      ;       necessary before any valid commands are sent.  The only
                          436      ;       exception is the force interrupt command, which naturally
                          437      ;       is sent when the device is busy.
                          438      ;
                          439      ;       Entry:  NONE
                          440      ;
                          441      ;       Exit:   NONE
                          442      ;
                          443      ;       Uses:   ???---(because CBA may use all)
                          444      ;
                          445
0077                      446      WNB     PROC    NEAR
                          447              PUBLIC  WNB
                          448
0077 E886FF               449              CALL    CSP
007A A801                 450              TEST    AL,TNS_BSY
007C 7407                 451              JZ      WNB1            ; Busy is de-asserted
                          452
007E E80000         E     453              CALL    CBA
0081 D0D8                 454              RCR     AL,1
0083 73F2                 455              JNC     WNB             ; The user has not aborted yet
                          456
0085 C3                   457      WNB1:   RET
                          458
                          459      WNB     ENDP
                          460
                          461
                          462
                          463
-----                     464      ASTLIB  ENDS
                          465
                          466              END
```

```
NAME            TYPE     VALUE  ATTRIBUTES, XREFS

EXPAND. . . . . V BYTE   0008H  S FIELD  80#
FD_CMND . . . . NUMBER   0000H    225#
FD_DATA . . . . NUMBER   0003H    228# 344
FD_SECT . . . . NUMBER   0002H    227#
FD_STAT . . . . NUMBER   0000H    224#
FD_TRCK . . . . NUMBER   0001H    226#
FONT. . . . . V DWORD  0000H  EXTRN  183#
FONTAB. . . . . SEGMENT        SIZE=0000H BYTE  PUBLIC 'DATA'  123# 125
FORE. . . . . V BYTE   0000H  S FIELD  48#
FUNCTION. . . . V WORD   0001H  S FIELD  64#
GRAPHIC . . . . V BYTE   0009H  S FIELD  81#
H19_MODE. . . . V   18  0000H  EXTRN  197#
H19_MODE_STRUC. STRUC          SIZE=0012H  #FIELDS=17  71 89# 197
H19_PAR . . . . V     9  0000H  EXTRN  198#
H19_PAR_STRUC . STRUC          SIZE=0009H  #FIELDS=9  91 101# 198
HORZ_CHAR . . . V BYTE   0000H  EXTRN  173#
INDEX . . . . . V BYTE   0000H  S FIELD  41#
INIT. . . . . L NEAR   0000H  EXTRN  165#
INSERT. . . . . V BYTE   000AH  S FIELD  82#
ITV_OBI . . . . V DWORD  0000H  EXTRN  132#
ITV_SST . . . . V DWORD  0000H  EXTRN  131#
KEY_EN. . . . . V BYTE   000BH  S FIELD  83#
LPS . . . . . V BYTE   0002H  S FIELD  94#
MUC . . . . . V DWORD  0000H  EXTRN  184#
MDL . . . . . V DWORD  0000H  EXTRN  185#
MODE. . . . . V BYTE   0003H  S FIELD  65#
MONENT. . . . . SEGMENT        SIZE=0000H PARA PUBLIC   136# 144
MSK . . . . . V BYTE   0002H  S FIELD  50#
MTR_DCRT. . . . L FAR    0000H  EXTRN  140#
MTR_DKBD. . . . L FAR    0000H  EXTRN  142#
MTR_FONT. . . . V BYTE   0000H  EXTRN  124#
MTR_MON . . . . L FAR    0000H  EXTRN  138#
MTR_RES . . . . L FAR    0000H  EXTRN  137#
MTR_SCRT. . . . L FAR    0000H  EXTRN  141#
MTR_SKBD. . . . L FAR    0000H  EXTRN  143#
MTR_SWIM. . . . L FAR    0000H  EXTRN  139#
MTR100. . . . . SEGMENT        SIZE=0000H PARA PUBLIC 'CODE'  145# 148 210
MTR101. . . . . L NEAR   0000H  EXTRN  154#
OPER_COUNT. . . V BYTE   0004H  S FIELD  66#
OPER_INDEX. . . V BYTE   0005H  S FIELD  67#
OPERAND . . . . V BYTE   0006H  S FIELD  68#
PAINTED . . . . V BYTE   0004H  S FIELD  52#
PFONT . . . . . V BYTE   0005H  S FIELD  53#
PLM_TRUE. . . . NUMBER   00FFH    29#
PORT. . . . . V BYTE   0001H  S FIELD  42# 256
POVRAM. . . . . V BYTE   0003H  S FIELD  95#
PROMPT. . . . . V DWORD  0000H  EXTRN  186#
PTR_DATA. . . . V DWORD  0004H  S FIELD  323# 343
RDC . . . . . V DWORD  0000H  EXTRN  187#
RDT1. . . . . L NEAR   002FH  ASTLIB  345# 348
RDTA. . . . . STRUC            SIZE=000EH  #FIELDS=6  320 327#
RDTRACKA. . . . L NEAR   0019H  ASTLIB PUBLIC  330# 331 354
REGP. . . . . V DWORD  0000H  EXTRN  175#
RESETF. . . . . V BYTE   0000H  EXTRN  176#
REVERSE . . . . V WORD   000CH  S FIELD  84#
```

```
SERIES-III 8086/8087/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE FONT
OBJECT MODULE PLACED IN :F2:FONT2.OBJ
INVOCATION LINE CONTROLS:  PRINT(:TO:) XREF ERRORPRINT(:TO:)


LOC  OBJ                  LINE     SOURCE
```

```
LOC  OBJ                  LINE     SOURCE

                            51     ;                '****    '
                            52     ;                '****    '
                            53     ;                '****    '
                            54     ;                '****    '
                            55     ;                '****    '
                            56     ;                '    ****'
                            57     ;                '    ****'
                            58     ;                '    ****'
                            59     ;                '    ****'
                            60     ;                '    ****'
                            61     ;                +--------+
                            62     ;
                            63     ;
                            64     ;       Basically, each of the characters consists of a ten
                            65     ;       element array which is to be displayed on the screen.
                            66     ;       Most of the work here was done by Steve Parker for
                            67     ;       the CP/M 8086 assembler.  Some of the descenders have
                            68     ;       been modified ('p', 'q', and 'y').  The format has been
                            69     ;       modified for Intel Assembler compatibility.
                            70     ;
                            71     ;                                                G. Chandler
                            72     ;
                            73     ;
                            74
                            75     %*DEFINE(GRAPHIC)(0)   %' No Graphic Characters
                            76     %*DEFINE(LOWER)(1)   %' Lower Case
                            77
----                        78     FONTAB  SEGMENT BYTE PUBLIC 'DATA'
                            79
                            80             NAME    FONT
                            81
0000                        82     MTR_FONT        LABEL   BYTE            ; Base Address of Font ROM
                            83
                            84             PUBLIC  MTR_FONT                ; Font ROM is a global
                            85
                            86 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

0030 04                    142              DB      00000100B
0031 08                    143              DB      00001000B
0032 10                    144              DB      00010000B
0033 26                    145              DB      00100110B
0034 06                    146              DB      00000110B
0035 00                    147              DB      00000000B
                           148      ;               "&"
0036 00                    149              DB      00000000B
0037 08                    150              DB      00001000B
0038 14                    151              DB      00010100B
0039 14                    152              DB      00010100B
003A 18                    153              DB      00011000B
003B 2A                    154              DB      00101010B
003C 24                    155              DB      00100100B
003D 1A                    156              DB      00011010B
003E 00                    157              DB      00000000B
                           158      ;               "'"
003F 00                    159              DB      00000000B
0040 0C                    160              DB      00001100B
0041 08                    161              DB      00001000B
0042 10                    162              DB      00010000B
0043 00                    163              DB      00000000B
0044 00                    164              DB      00000000B
0045 00                    165              DB      00000000B
0046 00                    166              DB      00000000B
0047 00                    167              DB      00000000B
                           168      ;               "("
0048 00                    169              DB      00000000B
0049 04                    170              DB      00000100B
004A 08                    171              DB      00001000B
004B 10                    172              DB      00010000B
004C 10                    173              DB      00010000B
004D 10                    174              DB      00010000B
004E 08                    175              DB      00001000B
004F 04                    176              DB      00000100B
0050 00                    177              DB      00000000B
                           178      ;               ")"
0051 00                    179              DB      00000000B
0052 10                    180              DB      00010000B
0053 08                    181              DB      00001000B
0054 04                    182              DB      00000100B
0055 04                    183              DB      00000100B
0056 04                    184              DB      00000100B
0057 08                    185              DB      00001000B
0058 10                    186              DB      00010000B
0059 00                    187              DB      00000000B
                           188      ;               "*"
005A 00                    189              DB      00000000B
005B 00                    190              DB      00000000B
005C 08                    191              DB      00001000B
005D 2A                    192              DB      00101010B
005E 1C                    193              DB      00011100B
005F 2A                    194              DB      00101010B
0060 08                    195              DB      00001000B
0061 00                    196              DB      00000000B
```

```
LOC  OBJ                  LINE     SOURCE

0093 26                    252              DB      00100110B
0094 2A                    253              DB      00101010B
0095 32                    254              DB      00110010B
0096 22                    255              DB      00100010B
0097 1C                    256              DB      00011100B
0098 00                    257              DB      00000000B
                           258      ;               "1"
0099 00                    259              DB      00000000B
009A 08                    260              DB      00001000B
009B 18                    261              DB      00011000B
009C 08                    262              DB      00001000B
009D 08                    263              DB      00001000B
009E 08                    264              DB      00001000B
009F 08                    265              DB      00001000B
00A0 1C                    266              DB      00011100B
00A1 00                    267              DB      00000000B
                           268      ;               "2"
00A2 00                    269              DB      00000000B
00A3 1C                    270              DB      00011100B
00A4 22                    271              DB      00100010B
00A5 02                    272              DB      00000010B
00A6 04                    273              DB      00000100B
00A7 08                    274              DB      00001000B
00A8 10                    275              DB      00010000B
00A9 3E                    276              DB      00111110B
00AA 00                    277              DB      00000000B
                           278      ;               "3"
00AB 00                    279              DB      00000000B
00AC 3E                    280              DB      00111110B
00AD 04                    281              DB      00000100B
00AE 08                    282              DB      00001000B
00AF 04                    283              DB      00000100B
00B0 02                    284              DB      00000010B
00B1 22                    285              DB      00100010B
00B2 1C                    286              DB      00011100B
00B3 00                    287              DB      00000000B
                           288      ;               "4"
00B4 00                    289              DB      00000000B
00B5 04                    290              DB      00000100B
00B6 0C                    291              DB      00001100B
00B7 14                    292              DB      00010100B
00B8 24                    293              DB      00100100B
00B9 3E                    294              DB      00111110B
00BA 04                    295              DB      00000100B
00BB 04                    296              DB      00000100B
00BC 00                    297              DB      00000000B
                           298      ;               "5"
00BD 00                    299              DB      00000000B
00BE 3E                    300              DB      00111110B
00BF 20                    301              DB      00100000B
00C0 3C                    302              DB      00111100B
00C1 02                    303              DB      00000010B
00C2 02                    304              DB      00000010B
00C3 22                    305              DB      00100010B
00C4 1C                    306              DB      00011100B
```

```
LOC  OBJ                  LINE     SOURCE

00F6 18                    362              DB      00011000B
00F7 00                    363              DB      00000000B
00F8 18                    364              DB      00011000B
00F9 18                    365              DB      00011000B
00FA 08                    366              DB      00001000B
00FB 10                    367              DB      00010000B
                           368      ;               "<"
00FC 00                    369              DB      00000000B
00FD 02                    370              DB      00000010B
00FE 04                    371              DB      00000100B
00FF 08                    372              DB      00001000B
0100 10                    373              DB      00010000B
0101 08                    374              DB      00001000B
0102 04                    375              DB      00000100B
0103 02                    376              DB      00000010B
0104 00                    377              DB      00000000B
                           378      ;               "="
0105 00                    379              DB      00000000B
0106 00                    380              DB      00000000B
0107 00                    381              DB      00000000B
0108 3E                    382              DB      00111110B
0109 00                    383              DB      00000000B
010A 3E                    384              DB      00111110B
010B 00                    385              DB      00000000B
010C 00                    386              DB      00000000B
010D 00                    387              DB      00000000B
                           388      ;               ">"
010E 00                    389              DB      00000000B
010F 20                    390              DB      00100000B
0110 10                    391              DB      00010000B
0111 08                    392              DB      00001000B
0112 04                    393              DB      00000100B
0113 08                    394              DB      00001000B
0114 10                    395              DB      00010000B
0115 20                    396              DB      00100000B
0116 00                    397              DB      00000000B
                           398      ;               "?"
0117 00                    399              DB      00000000B
0118 1C                    400              DB      00011100B
0119 22                    401              DB      00100010B
011A 02                    402              DB      00000010B
011B 04                    403              DB      00000100B
011C 08                    404              DB      00001000B
011D 00                    405              DB      00000000B
011E 08                    406              DB      00001000B
011F 00                    407              DB      00000000B
                           408      ;               "@"
0120 00                    409              DB      00000000B
0121 0C                    410              DB      00001100B
0122 12                    411              DB      00010010B
0123 26                    412              DB      00100110B
0124 2A                    413              DB      00101010B
0125 2E                    414              DB      00101110B
0126 20                    415              DB      00100000B
0127 1E                    416              DB      00011110B
```

```
LOC  OBJ                  LINE     SOURCE

0159 20                    472              DB      00100000B
015A 3C                    473              DB      00111100B
015B 20                    474              DB      00100000B
015C 20                    475              DB      00100000B
015D 20                    476              DB      00100000B
015E 00                    477              DB      00000000B
                           478      ;               "G"
015F 00                    479              DB      00000000B
0160 1C                    480              DB      00011100B
0161 22                    481              DB      00100010B
0162 20                    482              DB      00100000B
0163 26                    483              DB      00100110B
0164 22                    484              DB      00100010B
0165 22                    485              DB      00100010B
0166 1E                    486              DB      00011110B
0167 00                    487              DB      00000000B
                           488      ;               "H"
0168 00                    489              DB      00000000B
0169 22                    490              DB      00100010B
016A 22                    491              DB      00100010B
016B 22                    492              DB      00100010B
016C 3E                    493              DB      00111110B
016D 22                    494              DB      00100010B
016E 22                    495              DB      00100010B
016F 22                    496              DB      00100010B
0170 00                    497              DB      00000000B
                           498      ;               "I"
0171 00                    499              DB      00000000B
0172 1C                    500              DB      00011100B
0173 08                    501              DB      00001000B
0174 08                    502              DB      00001000B
0175 08                    503              DB      00001000B
0176 08                    504              DB      00001000B
0177 08                    505              DB      00001000B
0178 1C                    506              DB      00011100B
0179 00                    507              DB      00000000B
                           508      ;               "J"
017A 00                    509              DB      00000000B
017B 0E                    510              DB      00001110B
017C 04                    511              DB      00000100B
017D 04                    512              DB      00000100B
017E 04                    513              DB      00000100B
017F 04                    514              DB      00000100B
0180 24                    515              DB      00100100B
0181 18                    516              DB      00011000B
0182 00                    517              DB      00000000B
                           518      ;               "K"
0183 00                    519              DB      00000000B
0184 22                    520              DB      00100010B
0185 24                    521              DB      00100100B
0186 28                    522              DB      00101000B
0187 30                    523              DB      00110000B
0188 28                    524              DB      00101000B
0189 24                    525              DB      00100100B
018A 22                    526              DB      00100010B
```

```
LOC  OBJ                  LINE     SOURCE

01BC 22                    582              DB      00100010B
01BD 22                    583              DB      00100010B
01BE 2A                    584              DB      00101010B
01BF 24                    585              DB      00100100B
01C0 1A                    586              DB      00011010B
01C1 00                    587              DB      00000000B
                           588      ;               "R"
01C2 00                    589              DB      00000000B
01C3 3C                    590              DB      00111100B
01C4 22                    591              DB      00100010B
01C5 22                    592              DB      00100010B
01C6 3C                    593              DB      00111100B
01C7 28                    594              DB      00101000B
01C8 24                    595              DB      00100100B
01C9 22                    596              DB      00100010B
01CA 00                    597              DB      00000000B
                           598      ;               "S"
01CB 00                    599              DB      00000000B
01CC 1C                    600              DB      00011100B
01CD 22                    601              DB      00100010B
01CE 20                    602              DB      00100000B
01CF 1C                    603              DB      00011100B
01D0 02                    604              DB      00000010B
01D1 22                    605              DB      00100010B
01D2 1C                    606              DB      00011100B
01D3 00                    607              DB      00000000B
                           608      ;               "T"
01D4 00                    609              DB      00000000B
01D5 3E                    610              DB      00111110B
01D6 08                    611              DB      00001000B
01D7 08                    612              DB      00001000B
01D8 08                    613              DB      00001000B
01D9 08                    614              DB      00001000B
01DA 08                    615              DB      00001000B
01DB 08                    616              DB      00001000B
01DC 00                    617              DB      00000000B
                           618      ;               "U"
01DD 00                    619              DB      00000000B
01DE 22                    620              DB      00100010B
01DF 22                    621              DB      00100010B
01E0 22                    622              DB      00100010B
01E1 22                    623              DB      00100010B
01E2 22                    624              DB      00100010B
01E3 22                    625              DB      00100010B
01E4 1C                    626              DB      00011100B
01E5 00                    627              DB      00000000B
                           628      ;               "V"
01E6 00                    629              DB      00000000B
01E7 22                    630              DB      00100010B
01E8 22                    631              DB      00100010B
01E9 22                    632              DB      00100010B
01EA 14                    633              DB      00010100B
01EB 14                    634              DB      00010100B
01EC 08                    635              DB      00001000B
01ED 08                    636              DB      00001000B
```

```
LOC  OBJ                  LINE     SOURCE

021F 10                    692              DB      00010000B
0220 08                    693              DB      00001000B
0221 04                    694              DB      00000100B
0222 02                    695              DB      00000010B
0223 01                    696              DB      00000001B
0224 00                    697              DB      00000000B
                           698      ;               "]"
0225 00                    699              DB      00000000B
0226 38                    700              DB      00111000B
0227 08                    701              DB      00001000B
0228 08                    702              DB      00001000B
0229 08                    703              DB      00001000B
022A 08                    704              DB      00001000B
022B 08                    705              DB      00001000B
022C 38                    706              DB      00111000B
022D 00                    707              DB      00000000B
                           708      ;               "^"
022E 00                    709              DB      00000000B
022F 08                    710              DB      00001000B
0230 14                    711              DB      00010100B
0231 22                    712              DB      00100010B
0232 00                    713              DB      00000000B
0233 00                    714              DB      00000000B
0234 00                    715              DB      00000000B
0235 00                    716              DB      00000000B
0236 00                    717              DB      00000000B
                           718      ;               "_"
0237 00                    719              DB      00000000B
0238 00                    720              DB      00000000B
0239 00                    721              DB      00000000B
023A 00                    722              DB      00000000B
023B 00                    723              DB      00000000B
023C 00                    724              DB      00000000B
023D 00                    725              DB      00000000B
023E 7F                    726              DB      01111111B
023F 00                    727              DB      00000000B
                           728      ;               "`"
0240 00                    729              DB      00000000B
0241 18                    730              DB      00011000B
0242 08                    731              DB      00001000B
0243 04                    732              DB      00000100B
0244 00                    733              DB      00000000B
0245 00                    734              DB      00000000B
0246 00                    735              DB      00000000B
0247 00                    736              DB      00000000B
0248 00                    737              DB      00000000B
                           738
                           739      %IF(%LOWER
                           740 +1       1) THEN(
                           741      ;  "a"
                           742       DB 00000000B
                           743       DB 00000000B
                           744       DB 00000000B
                           745       DB 00011100B
                           746       DB 00000010B
```

```
LOC  OBJ                  LINE     SOURCE

                           802      DB 00000000B
                           803      DB 00000000B
                           804      DB 00000000B
                           805      DB 00011110B
                           806      DB 00100100B
                           807      DB 00111000B
                           808      DB 00011100B
                           809      DB 00100010B
                           810      DB 00011100B
                           811     ;  "h"
                           812      DB 00000000B
                           813      DB 00100000B
                           814      DB 00100000B
                           815      DB 00111100B
                           816      DB 00100010B
                           817      DB 00100010B
                           818      DB 00100010B
                           819      DB 00100010B
                           820      DB 00000000B
                           821     ;  "i"
                           822      DB 00000000B
                           823      DB 00001000B
                           824      DB 00000000B
                           825      DB 00011000B
                           826      DB 00001000B
                           827      DB 00001000B
                           828      DB 00001000B
                           829      DB 00011100B
                           830      DB 00000000B
                           831     ;  "j"
                           832      DB 00000000B
                           833      DB 00000010B
                           834      DB 00000000B
                           835      DB 00000010B
                           836      DB 00000010B
                           837      DB 00000010B
                           838      DB 00000010B
                           839      DB 00100010B
                           840      DB 00011100B
                           841     ;  "k"
                           842      DB 00000000B
                           843      DB 00100000B
                           844      DB 00100000B
                           845      DB 00100100B
                           846      DB 00101000B
                           847      DB 00110100B
                           848      DB 00100010B
                           849      DB 00100010B
                           850      DB 00000000B
                           851     ;  "l"
                           852      DB 00000000B
                           853      DB 00011000B
                           854      DB 00001000B
                           855      DB 00001000B
                           856      DB 00001000B
```

```
LOC  OBJ                  LINE     SOURCE

                           912      DB 00000000B
                           913      DB 00000000B
                           914      DB 00000000B
                           915      DB 00101100B
                           916      DB 00110010B
                           917      DB 00100000B
                           918      DB 00100000B
                           919      DB 00100000B
                           920      DB 00000000B
                           921     ;  "s"
                           922      DB 00000000B
                           923      DB 00000000B
                           924      DB 00000000B
                           925      DB 00011100B
                           926      DB 00100000B
                           927      DB 00011100B
                           928      DB 00000010B
                           929      DB 00011100B
                           930      DB 00000000B
                           931     ;  "t"
                           932      DB 00000000B
                           933      DB 00010000B
                           934      DB 00010000B
                           935      DB 00111000B
                           936      DB 00010000B
                           937      DB 00010000B
                           938      DB 00010010B
                           939      DB 00001100B
                           940      DB 00000000B
                           941     ;  "u"
                           942      DB 00000000B
                           943      DB 00000000B
                           944      DB 00000000B
                           945      DB 00100010B
                           946      DB 00100010B
                           947      DB 00100010B
                           948      DB 00100110B
                           949      DB 00011010B
                           950      DB 00000000B
                           951     ;  "v"
                           952      DB 00000000B
                           953      DB 00000000B
                           954      DB 00000000B
                           955      DB 00100010B
                           956      DB 00100010B
                           957      DB 00100010B
                           958      DB 00010100B
                           959      DB 00001000B
                           960      DB 00000000B
                           961     ;  "w"
                           962      DB 00000000B
                           963      DB 00000000B
                           964      DB 00000000B
                           965      DB 00100010B
                           966      DB 00100010B
```

```
LOC  OBJ                  LINE     SOURCE

                          1022       DB 00000000B
                          1023       DB 00011000B
                          1024       DB 00000100B
                          1025       DB 00000100B
                          1026       DB 00000010B
                          1027       DB 00000100B
                          1028       DB 00000100B
                          1029       DB 00011000B
                          1030       DB 00000000B
                          1031     ;  "~"
                          1032       DB 00000000B
                          1033       DB 00110000B
                          1034       DB 01001001B
                          1035       DB 00000110B
                          1036       DB 00000000B
                          1037       DB 00000000B
                          1038       DB 00000000B
                          1039       DB 00000000B
                          1040       DB 00000000B
                          1041     )FI
                          1042 +1
                          1043 +1  ;  "a"
0249 00                   1044 +1    DB 00000000B
024A 00                   1045 +1    DB 00000000B
024B 00                   1046 +1    DB 00000000B
024C 1C                   1047 +1    DB 00011100B
024D 02                   1048 +1    DB 00000010B
024E 1E                   1049 +1    DB 00011110B
024F 22                   1050 +1    DB 00100010B
0250 1E                   1051 +1    DB 00011110B
0251 00                   1052 +1    DB 00000000B
                          1053 +1  ;  "b"
0252 00                   1054 +1    DB 00000000B
0253 20                   1055 +1    DB 00100000B
0254 20                   1056 +1    DB 00100000B
0255 3C                   1057 +1    DB 00111100B
0256 22                   1058 +1    DB 00100010B
0257 22                   1059 +1    DB 00100010B
0258 22                   1060 +1    DB 00100010B
0259 3C                   1061 +1    DB 00111100B
025A 00                   1062 +1    DB 00000000B
                          1063 +1  ;  "c"
025B 00                   1064 +1    DB 00000000B
025C 00                   1065 +1    DB 00000000B
025D 00                   1066 +1    DB 00000000B
025E 1C                   1067 +1    DB 00011100B
025F 22                   1068 +1    DB 00100010B
0260 20                   1069 +1    DB 00100000B
0261 20                   1070 +1    DB 00100000B
0262 1C                   1071 +1    DB 00011100B
0263 00                   1072 +1    DB 00000000B
                          1073 +1  ;  "d"
0264 00                   1074 +1    DB 00000000B
0265 02                   1075 +1    DB 00000010B
0266 02                   1076 +1    DB 00000010B
```

```
LOC  OBJ                  LINE      SOURCE

0299 00                   1132 +1    DB 00000000B
                          1133 +1   ;  "j"
029A 00                   1134 +1    DB 00000000B
029B 02                   1135 +1    DB 00000010B
029C 00                   1136 +1    DB 00000000B
029D 02                   1137 +1    DB 00000010B
029E 02                   1138 +1    DB 00000010B
029F 02                   1139 +1    DB 00000010B
02A0 02                   1140 +1    DB 00000010B
02A1 22                   1141 +1    DB 00100010B
02A2 1C                   1142 +1    DB 00011100B
                          1143 +1   ;  "k"
02A3 00                   1144 +1    DB 00000000B
02A4 20                   1145 +1    DB 00100000B
02A5 20                   1146 +1    DB 00100000B
02A6 24                   1147 +1    DB 00100100B
02A7 28                   1148 +1    DB 00101000B
02A8 34                   1149 +1    DB 00110100B
02A9 22                   1150 +1    DB 00100010B
02AA 22                   1151 +1    DB 00100010B
02AB 00                   1152 +1    DB 00000000B
                          1153 +1   ;  "l"
02AC 00                   1154 +1    DB 00000000B
02AD 18                   1155 +1    DB 00011000B
02AE 08                   1156 +1    DB 00001000B
02AF 08                   1157 +1    DB 00001000B
02B0 08                   1158 +1    DB 00001000B
02B1 08                   1159 +1    DB 00001000B
02B2 08                   1160 +1    DB 00001000B
02B3 1C                   1161 +1    DB 00011100B
02B4 00                   1162 +1    DB 00000000B
                          1163 +1   ;  "m"
02B5 00                   1164 +1    DB 00000000B
02B6 00                   1165 +1    DB 00000000B
02B7 00                   1166 +1    DB 00000000B
02B8 34                   1167 +1    DB 00110100B
02B9 2A                   1168 +1    DB 00101010B
02BA 2A                   1169 +1    DB 00101010B
02BB 2A                   1170 +1    DB 00101010B
02BC 2A                   1171 +1    DB 00101010B
02BD 00                   1172 +1    DB 00000000B
                          1173 +1   ;  "n"
02BE 00                   1174 +1    DB 00000000B
02BF 00                   1175 +1    DB 00000000B
02C0 00                   1176 +1    DB 00000000B
02C1 3C                   1177 +1    DB 00111100B
02C2 22                   1178 +1    DB 00100010B
02C3 22                   1179 +1    DB 00100010B
02C4 22                   1180 +1    DB 00100010B
02C5 22                   1181 +1    DB 00100010B
02C6 00                   1182 +1    DB 00000000B
                          1183 +1   ;  "o"
02C7 00                   1184 +1    DB 00000000B
02C8 00                   1185 +1    DB 00000000B
02C9 00                   1186 +1    DB 00000000B
```

```
LOC  OBJ                  LINE     SOURCE

02FC 00                   1242 +1   DB 00000000B
                          1243 +1  ;  "u"
02FD 00                   1244 +1   DB 00000000B
02FE 00                   1245 +1   DB 00000000B
02FF 00                   1246 +1   DB 00000000B
0300 22                   1247 +1   DB 00100010B
0301 22                   1248 +1   DB 00100010B
0302 22                   1249 +1   DB 00100010B
0303 26                   1250 +1   DB 00100110B
0304 1A                   1251 +1   DB 00011010B
0305 00                   1252 +1   DB 00000000B
                          1253 +1  ;  "v"
0306 00                   1254 +1   DB 00000000B
0307 00                   1255 +1   DB 00000000B
0308 00                   1256 +1   DB 00000000B
0309 22                   1257 +1   DB 00100010B
030A 22                   1258 +1   DB 00100010B
030B 22                   1259 +1   DB 00100010B
030C 14                   1260 +1   DB 00010100B
030D 08                   1261 +1   DB 00001000B
030E 00                   1262 +1   DB 00000000B
                          1263 +1  ;  "w"
030F 00                   1264 +1   DB 00000000B
0310 00                   1265 +1   DB 00000000B
0311 00                   1266 +1   DB 00000000B
0312 22                   1267 +1   DB 00100010B
0313 22                   1268 +1   DB 00100010B
0314 2A                   1269 +1   DB 00101010B
0315 2A                   1270 +1   DB 00101010B
0316 14                   1271 +1   DB 00010100B
0317 00                   1272 +1   DB 00000000B
                          1273 +1  ;  "x"
0318 00                   1274 +1   DB 00000000B
0319 00                   1275 +1   DB 00000000B
031A 00                   1276 +1   DB 00000000B
031B 22                   1277 +1   DB 00100010B
031C 14                   1278 +1   DB 00010100B
031D 08                   1279 +1   DB 00001000B
031E 14                   1280 +1   DB 00010100B
031F 22                   1281 +1   DB 00100010B
0320 00                   1282 +1   DB 00000000B
                          1283 +1  ;  "y"
0321 00                   1284 +1   DB 00000000B
0322 00                   1285 +1   DB 00000000B
0323 00                   1286 +1   DB 00000000B
0324 22                   1287 +1   DB 00100010B
0325 22                   1288 +1   DB 00100010B
0326 22                   1289 +1   DB 00100010B
0327 1E                   1290 +1   DB 00011110B
0328 02                   1291 +1   DB 00000010B
0329 1C                   1292 +1   DB 00011100B
                          1293 +1  ;  "z"
032A 00                   1294 +1   DB 00000000B
032B 00                   1295 +1   DB 00000000B
032C 00                   1296 +1   DB 00000000B
```

```
LOC  OBJ                  LINE     SOURCE

                          1352      DB 00111110B
                          1353      DB 00111110B
                          1354      DB 00011100B
                          1355      DB 00000000B
                          1356      DB 00000000B
                          1357     ; graphic "_"
                          1358      DB 11111111B
                          1359      DB 01111111B
                          1360      DB 00111111B
                          1361      DB 00011111B
                          1362      DB 00001111B
                          1363      DB 00000111B
                          1364      DB 00000011B
                          1365      DB 00000001B
                          1366      DB 00000000B
                          1367     ; graphic "`"
                          1368      DB 00011000B
                          1369      DB 00011000B
                          1370      DB 00011000B
                          1371      DB 00011000B
                          1372      DB 00011000B
                          1373      DB 00011000B
                          1374      DB 00011000B
                          1375      DB 00011000B
                          1376      DB 00011000B
                          1377     ; graphic "a"
                          1378      DB 00000000B
                          1379      DB 00000000B
                          1380      DB 00000000B
                          1381      DB 00000000B
                          1382      DB 11111111B
                          1383      DB 00000000B
                          1384      DB 00000000B
                          1385      DB 00000000B
                          1386      DB 00000000B
                          1387     ; graphic "b"
                          1388      DB 00011000B
                          1389      DB 00011000B
                          1390      DB 00011000B
                          1391      DB 00011000B
                          1392      DB 11111111B
                          1393      DB 00011000B
                          1394      DB 00011000B
                          1395      DB 00011000B
                          1396      DB 00011000B
                          1397     ; graphic "c"
                          1398      DB 00000000B
                          1399      DB 00000000B
                          1400      DB 00000000B
                          1401      DB 00000000B
                          1402      DB 11111000B
                          1403      DB 00011000B
                          1404      DB 00011000B
                          1405      DB 00011000B
                          1406      DB 00011000B
```

```
LOC  OBJ                  LINE     SOURCE

                          1462      DB 10101010B
                          1463      DB 01010101B
                          1464      DB 10101010B
                          1465      DB 01010101B
                          1466      DB 10101010B
                          1467     ; graphic "j"
                          1468     %IF(%NES(0,1))
                          1469       THEN (
                          1470      DB 11110000B
                          1471      DB 11110000B
                          1472      DB 11110000B
                          1473      DB 11110000B
                          1474      DB 11111111B
                          1475      DB 00001111B
                          1476      DB 00001111B
                          1477      DB 00001111B
                          1478      DB 00001111B )
                          1479       ELSE (
                          1480      DB 00000000B
                          1481      DB 00000000B
                          1482      DB 00001000B
                          1483      DB 00000000B
                          1484      DB 00111110B
                          1485      DB 00000000B
                          1486      DB 00001000B
                          1487      DB 00000000B
                          1488      DB 00000000B )FI
                          1489     ; graphic "k"
                          1490      DB 00000000B
                          1491      DB 00000000B
                          1492      DB 00001000B
                          1493      DB 00001000B
                          1494      DB 00101010B
                          1495      DB 00011100B
                          1496      DB 00001000B
                          1497      DB 00000000B
                          1498      DB 00000000B
                          1499     ; graphic "l"
                          1500      DB 00000000B
                          1501      DB 00000000B
                          1502      DB 00000000B
                          1503      DB 00000000B
                          1504      DB 00001111B
                          1505      DB 00001111B
                          1506      DB 00001111B
                          1507      DB 00001111B
                          1508      DB 00001111B
                          1509     ; graphic "m"
                          1510      DB 00000000B
                          1511      DB 00000000B
                          1512      DB 00000000B
                          1513      DB 00000000B
                          1514      DB 11110000B
                          1515      DB 11110000B
                          1516      DB 11110000B
```

```
LOC  OBJ                  LINE     SOURCE

                          1572      DB 00000000B
                          1573      DB 00000000B
                          1574      DB 11111111B
                          1575      DB 00011000B
                          1576      DB 00011000B
                          1577      DB 00011000B
                          1578      DB 00011000B
                          1579     ; graphic "t"
                          1580      DB 00011000B
                          1581      DB 00011000B
                          1582      DB 00011000B
                          1583      DB 00011000B
                          1584      DB 11111000B
                          1585      DB 00011000B
                          1586      DB 00011000B
                          1587      DB 00011000B
                          1588      DB 00011000B
                          1589     ; graphic "u"
                          1590      DB 00011000B
                          1591      DB 00011000B
                          1592      DB 00011000B
                          1593      DB 00011000B
                          1594      DB 11111111B
                          1595      DB 00000000B
                          1596      DB 00000000B
                          1597      DB 00000000B
                          1598      DB 00000000B
                          1599     ; graphic "v"
                          1600      DB 00011000B
                          1601      DB 00011000B
                          1602      DB 00011000B
                          1603      DB 00011000B
                          1604      DB 00011111B
                          1605      DB 00011000B
                          1606      DB 00011000B
                          1607      DB 00011000B
                          1608      DB 00011000B
                          1609     ; graphic "w"
                          1610      DB 10000001B
                          1611      DB 11000011B
                          1612      DB 01100110B
                          1613      DB 00111100B
                          1614      DB 00011000B
                          1615      DB 00111100B
                          1616      DB 01100110B
                          1617      DB 11000011B
                          1618      DB 10000001B
                          1619     ; graphic "x"
                          1620      DB 00000001B
                          1621      DB 00000011B
                          1622      DB 00000110B
                          1623      DB 00001100B
                          1624      DB 00011000B
                          1625      DB 00110000B
                          1626      DB 01100000B
```

```
LOC  OBJ                  LINE     SOURCE

                          1682       DB 00111100B
                          1683       DB 00111100B
                          1684       DB 00011100B
                          1685       DB 00001100B
                          1686       DB 00001100B
                          1687       DB 00000000B
                          1688       DB 00000000B
                          1689     )FI
                          1690 +1
                          1691
-----                     1692     FONTAB  ENDS
                          1693
                          1694             END
```

```
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:GLOBAL.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:I85DEF.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:INIT.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:IODEF.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:ITCDEF.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:KEYDEF.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:KEYLIB.H')
=    $SAVE
=    $NOLIST

     $       INCLUDE (':F2:MISC.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:MTR100.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:P6821.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:REGDEF.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:VIDEO.H')
=    $SAVE
=    $NOLIST


     $EJECT
```

```
306   1       COMMAND:          PROCEDURE                EXTERNAL;
307   2         END;

308   1       C_BOOT:           PROCEDURE                EXTERNAL;
309   2         END;

310   1       INIT:             PROCEDURE                EXTERNAL;
311   2         END;

                $       IF EXTENDED_MONITOR
                SSTEP:            PROCEDURE              EXTERNAL;
                  END;
                $       ENDIF

312   1         DECLARE       STEP_COUNT      WORD EXTERNAL;

                $SUBTITLE('MTR-101      - Monitor Loop')
```

```
323   1     VIDEO_INTR:     PROCEDURE       PUBLIC;
324   2       DECLARE       TEMP            BYTE;
325   2         DO;
326   3         CALL SDS;                   /* Set the Data Segment         */

                /*
                 *  Vertical Retrace Interrupt
                 */
327   3         IF  ( INPUT(GEN_CNT) AND P6821_IRQ2 ) <> 0  THEN
328   3           DO;
329   4           TEMP=INPUT(GEN_DAT);      /* Clear Interrupt              */
330   4           OUTPUT(GEN_DAT)=(INPUT(GEN_DAT) AND (NOT 0010$0000B));
331   4           TEMP=INPUT(IO_DIP);       /* Dummy I/O for 6821           */
332   4           OUTPUT(GEN_DAT)=(INPUT(GEN_DAT) OR 0010$0000B);
333   4           CALL MTR_TTY_INTR;        /* Vertical Retrace             */
334   4           END;

                /*
                 *  Key-Board Interrupt
                 */
335   3         IF  (INPUT(KEY_STAT) AND KS_CXRDY) <> 0  THEN
336   3           DO;
337   4           KEY.CHAR=INPUT(KEY_DATA);
338   4           KEY.AVAIL=TRUE;
339   4           END;

340   3         END;
341   2       END;


              $EJECT
```

```
                                        ; STATEMENT # 6
                                        ; STATEMENT # 20
                                        ; STATEMENT # 39
                                        ; STATEMENT # 53
                                        ; STATEMENT # 67
                                        ; STATEMENT # 108
                                        ; STATEMENT # 109
                                        ; STATEMENT # 111
                                        ; STATEMENT # 138
                                        ; STATEMENT # 193
                                        ; STATEMENT # 229
                                        ; STATEMENT # 237
                                        ; STATEMENT # 243
                                        ; STATEMENT # 255
                                        ; STATEMENT # 265
                                        ; STATEMENT # 275
                                        ; STATEMENT # 304
                                        ; STATEMENT # 314
          MTR101        PROC NEAR
0024  55               PUSH    BP
0025  8BEC             MOV     BP,SP
                                        ; STATEMENT # 315
0027  B007             MOV     AL,7H
0029  50               PUSH    AX      ; 1
002A  E80000           CALL    WRITEC
                                        ; STATEMENT # 316
002D  A00000           MOV     AL,RESETF
0030  D0D8             RCR     AL,1
0032  7309             JNB     @1
0034  E4FF             IN      0FFH
0036  A808             TEST    AL,8H
0038  7403             JZ      @1
                                        ; STATEMENT # 317
003A  E80000           CALL    C_BOOT
                                        ; STATEMENT # 318
          @1:
003D  C606000000       MOV     RESETF,0H
          @2:
                                        ; STATEMENT # 320
0042  E80000           CALL    COMMAND
                                        ; STATEMENT # 321
0045  EBFB             JMP     @2
                                        ; STATEMENT # 322
          MTR101        ENDP
                                        ; STATEMENT # 323
          VIDEO_INTR        PROC NEAR
0047  55               PUSH    BP
0048  8BEC             MOV     BP,SP
                                        ; STATEMENT # 326
004A  E80000           CALL    SDS
                                        ; STATEMENT # 327
004D  E4E1             IN      0E1H
004F  A840             TEST    AL,40H
0051  741B             JZ      @4
                                        ; STATEMENT # 329
0053  E4E0             IN      0E0H
```

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES	
291	0000H	1	BACK	BYTE IN PROC (SCA) PARAMETER 291
7			BEL.	LITERALLY '007H' 315
94	0000H	1	BIIL	BYTE EXTERNAL(36)
4			BOOL	LITERALLY 'BYTE' 98 103 104 106 108 230 231 250 285
74	0000H	83	BOOT_PAR	STRUCTURE EXTERNAL(16)
	0000H	1	INDEX.	BYTE
	0001H	1	PORT	BYTE
	0002H	80	STRNG.	BYTE ARRAY(80)
	0052H	1	UNIT	BYTE
8			BS	LITERALLY '008H'
95	0000H	1	BSIL	BYTE EXTERNAL(37)
27	0000H	4	BYTEPTR.	POINTER IN PROC (INCB) PARAMETER 27
30	0000H	1	C.	BYTE IN PROC (SXMTC) PARAMETER 30
22	0000H	1	C.	BYTE IN PROC (DXMTC) PARAMETER 22
9			CAN.	LITERALLY '018H'
279	0000H	1	CHAR	BYTE IN PROC (D_CRT) PARAMETER 279
58	0000H	1	CHAR	BYTE IN PROC (MCU) PARAMETER 58
63	0000H	1	CHAR	BYTE IN PROC (WRITEC) PARAMETER 63
236	0000H	1	CMND	BYTE IN PROC (WRITK) PARAMETER 236
68			CODE_SEG	LITERALLY '0FE05H'
303	0000H	1	COL.	BYTE IN PROC (XMTSC) PARAMETER 303
102	0000H	7	COLOR.	STRUCTURE EXTERNAL(41)
	0000H	1	FORE	BYTE
	0001H	1	BACK	BYTE
	0002H	1	MASK	BYTE
	0003H	1	CLEAR.	BYTE
	0004H	1	PAINTED. . . .	BYTE
	0005H	1	FONT	BYTE
	0006H	1	COMP_FONT. . .	BYTE
306	0000H		COMMAND.	PROCEDURE EXTERNAL(71) STACK=0000H 320
313	0000H	35	COPYRIGHT. . . .	BYTE ARRAY(35) DATA
303	0000H	2	COUNT.	WORD IN PROC (XMTSC) PARAMETER 303
273			CPU_AF	LITERALLY '00000000$00010000B'
275			CPU_CF	LITERALLY '00000000$00000001B'
268			CPU_DF	LITERALLY '00000100$00000000B'
269			CPU_IF	LITERALLY '00000010$00000000B'
267			CPU_OF	LITERALLY '00001000$00000000B'
274			CPU_PF	LITERALLY '00000000$00000100B'
271			CPU_SF	LITERALLY '00000000$10000000B'
270			CPU_TF	LITERALLY '00000001$00000000B'
272			CPU_ZF	LITERALLY '00000000$01000000B'
10			CR	LITERALLY '00DH' 313
55	0000H		CRLF	PROCEDURE EXTERNAL(8) STACK=0000H
103	0000H	3	CRTC_CURSOR. . .	STRUCTURE EXTERNAL(42)
	0000H	2	START.	WORD
	0002H	1	UPDATE	BYTE
104	0000H	3	CRTC_DISPLAY . .	STRUCTURE EXTERNAL(43)
	0000H	2	START.	WORD
	0002H	1	UPDATE	BYTE
276	0000H		CURSOR	PROCEDURE EXTERNAL(59) STACK=0000H
308	0000H		C_BOOT	PROCEDURE EXTERNAL(72) STACK=0000H 317

```
 15                    FF . . . . . .    LITERALLY '00CH'
 24  0000H             FLAGS. . . . . .  PROCEDURE WORD EXTERNAL(1) STACK=0000H
 80  0000H      4      FONT . . . . . .  POINTER EXTERNAL(22)
 89  0000H      2      FONTSIZE . . . . .  WORD EXTERNAL(31)
291  0000H      1      FORE . . . . . .  BYTE IN PROC (SCA) PARAMETER       291
241                    GEN_CNT. . . . . .  LITERALLY 'IO_GENERAL+1'          327
242                    GEN_DAT. . . . . .  LITERALLY 'IO_GENERAL'       329  330  332
243                    GEN_DIR. . . . . .  LITERALLY 'IO_GENERAL'
106  0000H     18      H19_MODE . . . . .  STRUCTURE EXTERNAL(45)
     0000H      1          ALTERNATE. . .  BYTE
     0001H      1          ANSI . . . . .  BYTE
     0002H      1          AUTO_CR. . . .  BYTE
     0003H      1          AUTO_LF. . . .  BYTE
     0004H      1          AUTO_REPEAT. .  BYTE
     0005H      1          BWO. . . . . .  BYTE
     0006H      1          CURSOR . . . .  BYTE
     0007H      1          CURSOR_ON. . .  BYTE
     0008H      1          EXPAND . . . .  BYTE
     0009H      1          GRAPHIC. . . .  BYTE
     000AH      1          INSERT . . . .  BYTE
     000BH      1          KEY_EN . . . .  BYTE
     000CH      2          REVERSE. . . .  WORD
     000EH      1          SHIFTED. . . .  BYTE
     000FH      1          STATUS . . . .  BYTE
     0010H      1          UPDN . . . . .  BYTE
     0011H      1          WRAP . . . . .  BYTE
107  0000H      9      H19_PAR. . . . . .  STRUCTURE EXTERNAL(46)
     0000H      1          CPL. . . . . .  BYTE
     0001H      1          DSC. . . . . .  BYTE
     0002H      1          LPS. . . . . .  BYTE
     0003H      1          POVRAM . . . .  BYTE
     0004H      1          SLI. . . . . .  BYTE
     0005H      1          SPC. . . . . .  BYTE
     0006H      1          SW401. . . . .  BYTE
     0007H      1          SW402. . . . .  BYTE
     0008H      1          VRAM_SIZE. . .  BYTE
 54  0000H                 HEX_DIGIT. . . .  BYTE ARRAY(0) EXTERNAL(7) DATA
 92  0000H      1      HORZ_CHAR. . . . .  BYTE EXTERNAL(34)
 16                    HT . . . . . .    LITERALLY '009H'
109                    I85_JMP. . . . .  LITERALLY '0303Q'
153                    ICW1_A7A5. . . .  LITERALLY '1110$0000B'
156                    ICW1_ADI4. . . .  LITERALLY '0000$0100B'
158                    ICW1_IC4 . . . .  LITERALLY '0000$0001B'
154                    ICW1_ICW1. . . .  LITERALLY '0001$0000B'
155                    ICW1_LTIM. . . .  LITERALLY '0000$1000B'
157                    ICW1_SNGL. . . .  LITERALLY '0000$0010B'
159                    ICW2_A15A8 . . .  LITERALLY '1111$1111B'
167                    ICW3_S0. . . . .  LITERALLY '0000$0001B'
166                    ICW3_S1. . . . .  LITERALLY '0000$0010B'
165                    ICW3_S2. . . . .  LITERALLY '0000$0100B'
164                    ICW3_S3. . . . .  LITERALLY '0000$1000B'
163                    ICW3_S4. . . . .  LITERALLY '0001$0000B'
162                    ICW3_S5. . . . .  LITERALLY '0010$0000B'
161                    ICW3_S6. . . . .  LITERALLY '0100$0000B'
160                    ICW3_S7. . . . .  LITERALLY '1000$0000B'
172                    ICW4_86. . . . .  LITERALLY '0000$0001B'
```

```
  205                    KC_KCO . . . . . .   LITERALLY '03H'
  210                    KC_MNS . . . . . .   LITERALLY '0BH'
  211                    KC_MUD . . . . . .   LITERALLY '0AH'
  212                    KC_RES . . . . . .   LITERALLY '00H'
  230  0000H        2    KEY. . . . . . . .   STRUCTURE EXTERNAL(49)
       0000H        1        CHAR . . . . .   BYTE        337*
       0001H        1        AVAIL. . . . .   BYTE        338*
  194                    KEY_CMND . . . . .   LITERALLY 'IO_KEYBRD+1'
  195                    KEY_DATA . . . . .   LITERALLY 'IO_KEYBRD+0'       337
  196                    KEY_STAT . . . . .   LITERALLY 'IO_KEYBRD+1'       335
   90  0000H      256    KMAP . . . . . . .   BYTE ARRAY(256) EXTERNAL(32)
  197                    KS_CXRDY . . . . .   LITERALLY '001B'        335
  198                    KS_RXRDY . . . . .   LITERALLY '010B'
  224                    KY_BREAK . . . . .   LITERALLY '0AAH'
  220                    KY_DOWN. . . . . .   LITERALLY '0A6H'
  213                    KY_ENTER . . . . .   LITERALLY '0DH'
  215                    KY_F0. . . . . . .   LITERALLY '096H'
  216                    KY_F12 . . . . . .   LITERALLY '0A2H'
  214                    KY_HELP. . . . . .   LITERALLY '095H'
  223                    KY_HOME. . . . . .   LITERALLY '0A9H'
  217                    KY_ID_CHAR . . . .   LITERALLY '0A3H'
  218                    KY_ID_LINE . . . .   LITERALLY '0A4H'
  227                    KY_KP_0. . . . . .   LITERALLY '0B0H'
  228                    KY_KP_9. . . . . .   LITERALLY '0B9H'
  225                    KY_KP_MINUS. . . .   LITERALLY '0ADH'
  226                    KY_KP_PERIOD . . .   LITERALLY '0AEH'
  222                    KY_LEFT. . . . . .   LITERALLY '0A8H'
  221                    KY_RIGHT . . . . .   LITERALLY '0A7H'
  229                    KY_SHIFT_OFFSET. .   LITERALLY '040H'
  219                    KY_UP. . . . . . .   LITERALLY '0A5H'
   17                    LF . . . . . . . .   LITERALLY '00AH'        313
  288  0000H        1    LINES. . . . . . .   BYTE IN PROC (ENABLE_LINES) PARAMETER          288
    3                    LT . . . . . . . .   LITERALLY 'LITERALLY'          4    5    6    7    8    9   10
                                                 11   12   13   14   15   16   17   18   19   20   40   41
                                                 42   43   44   45   46   47   48   49   50   51   52   53
                                                 68   69   70   99  100  101  109  112  113  114  115  116
                                                117  118  119  120  121  122  123  124  125  126  127  128
                                                129  130  131  132  133  134  135  136  137  138  139  140
                                                141  142  143  144  145  146  147  148  149  150  151  152
                                                153  154  155  156  157  158  159  160  161  162  163  164
                                                165  166  167  168  169  170  171  172  173  174  175  176
                                                177  178  179  180  181  182  183  184  185  186  187  188
                                                189  190  191  192  193  194  195  196  197  198  199  200
                                                201  202  203  204  205  206  207  208  209  210  211  212
                                                213  214  215  216  217  218  219  220  221  222  223  224
                                                225  226  227  228  229  238  239  240  241  242  243  256
                                                257  258  259  260  261  262  263  264  265  267  268  269
                                                270  271  272  273  274  275
  296  0000H        1    MASK . . . . . . .   BYTE IN PROC (SCP1) PARAMETER        296
   57  0000H             MCU. . . . . . . .   PROCEDURE BYTE EXTERNAL(9) STACK=0000H
   81  0000H        4    MDC. . . . . . . .   POINTER EXTERNAL(23)
   82  0000H        4    MDL. . . . . . . .   POINTER EXTERNAL(24)
  314  0024H       35    MTR101 . . . . . .   PROCEDURE PUBLIC STACK=0006H
       0024H             MTR101_. . . . . .   PROCEDURE STACK=0000H
  252  0000H             MTR_IRET . . . . .   PROCEDURE EXTERNAL(57) STACK=0000H
   70                    MTR_ISSUE. . . . .   LITERALLY '54H'
```

```
      000CH     2       SP . . . . . .    WORD
      000EH     2       DX . . . . . .    WORD
      0010H     2       CX . . . . . .    WORD
      0012H     2       BX . . . . . .    WORD
      0014H     2       AX . . . . . .    WORD
      0016H     2       IP . . . . . .    WORD
      0018H     2       CS . . . . . .    WORD
      001AH     2       FLAGS. . . . .    WORD
 35   0000H     4   REGP . . . . . .      POINTER IN PROC (XEC) PARAMETER          35
 97   0000H     4   REGP . . . . . .      POINTER EXTERNAL(39)
 98   0000H     1   RESETF . . . . . .    BYTE EXTERNAL(40)       316  318*
300   0000H         REV_SCROLL . . . .    PROCEDURE EXTERNAL(69) STACK=0000H
303   0000H     1   ROW. . . . . .        BYTE IN PROC (XMTSC) PARAMETER     303
290   0000H         SCA. . . . . .        PROCEDURE EXTERNAL(65) STACK=0000H
293   0000H         SCP. . . . . .        PROCEDURE EXTERNAL(66) STACK=0000H
295   0000H         SCP1 . . . . . .      PROCEDURE EXTERNAL(67) STACK=0000H
298   0000H         SCROLL . . . . . .    PROCEDURE EXTERNAL(68) STACK=0000H
254   0000H         SDS. . . . . .        PROCEDURE EXTERNAL(58) STACK=0000H       326
312   0000H     2   STEP_COUNT . . . .    WORD EXTERNAL(74)
 66   0000H     4   STRNGP . . . . . .    POINTER IN PROC (WRITES) PARAMETER        66
 29   0000H         SXMTC. . . . . .      PROCEDURE EXTERNAL(3) STACK=0000H
140               S_INT0 . . . . . .      LITERALLY '24'
151               S_ITC_0. . . . . .      LITERALLY 'IO_INT_SL+0'
152               S_ITC_1. . . . . .      LITERALLY 'IO_INT_SL+1'
 85   0000H     4   S_XMTC . . . . . .    POINTER EXTERNAL(27)
324   0007H     1   TEMP . . . . . .      BYTE IN PROC (VIDEO_INTR)          329*  331*
231   0000H         TESTK. . . . . .      PROCEDURE BYTE EXTERNAL(50) STACK=0000H
  6               TRUE . . . . . .        LITERALLY '0FFH'        338
283   0000H         TTY_INTR . . . . .    PROCEDURE EXTERNAL(62) STACK=0000H
285   0000H         TTY_POLL . . . . .    PROCEDURE BYTE EXTERNAL(63) STACK=0000H
 86   0000H     4   UIES . . . . . .      POINTER EXTERNAL(28)
296   0000H     1   VALU . . . . . .      BYTE IN PROC (SCP1) PARAMETER      296
 88   0000H    52   VECTORS. . . . . .    POINTER ARRAY(13) EXTERNAL(30)
 72   0000H     1   VERSION. . . . . .    BYTE EXTERNAL(14)
 93   0000H     1   VERT_LINE. . . . .    BYTE EXTERNAL(35)
323   0047H    57   VIDEO_INTR . . . .    PROCEDURE PUBLIC STACK=0006H
 32   0000H         VIDEO_INTR_A . . .    PROCEDURE EXTERNAL(4) STACK=0000H
 18               VT . . . . . .          LITERALLY '00BH'
 71   0000H     5   WIP. . . . . .        BYTE ARRAY(5) EXTERNAL(13)
 62   0000H         WRITEC . . . . . .    PROCEDURE EXTERNAL(11) STACK=0000H       315
 65   0000H         WRITES . . . . . .    PROCEDURE EXTERNAL(12) STACK=0000H
235   0000H         WRITK. . . . . .      PROCEDURE EXTERNAL(52) STACK=0000H
 87   0000H     4   XCA. . . . . .        POINTER EXTERNAL(29)
101               XCOLOR_BACK. . . .      LITERALLY '00$111$000B'
100               XCOLOR_FORE. . . .      LITERALLY '00$000$111B'
 34   0000H         XEC. . . . . .        PROCEDURE EXTERNAL(5) STACK=0000H
108   0000H    10   XMT. . . . . .        STRUCTURE EXTERNAL(47)
      0000H     1       BURST. . . . .    BYTE
      0001H     2       BCOUNT . . . .    INTEGER
      0003H     2       COUNT. . . . .    WORD
      0005H     1       COL. . . . . .    BYTE
      0006H     1       ROW. . . . . .    BYTE
      0007H     1       COLOR. . . . .    BYTE
      0008H     1       GRAPHIC. . . .    BYTE
      0009H     1       REVERSE. . . .    BYTE
302   0000H         XMTSC. . . . . .      PROCEDURE EXTERNAL(70) STACK=0000H
```

```
SERIES-III PL/M-86 V2.0 COMPILATION OF MODULE COMMND
OBJECT MODULE PLACED IN :F2:COMMND.OBJ
COMPILER INVOKED BY:  PLM86.86 :F2:COMMND.PLM CODE PRINT(:TO:)


                $TITLE('MTR-100 Z-Machine Monitor ROM:  Command Processor')
                $SUBTITLE('Compiler Controls')
                $       INCLUDE (':F2:COMCON.H')
            =   $SUBTITLE('Compiler Controls')
            =   $OPTIMIZE(3)
            =   $NOOVERFLOW
            =   $COMPACT
            =   $ROM
            =   $XREF
            =   $LARGE( MTR100 EXPORTS  MTR_MON;
            =   $   EXPORTS     MTR_MON;
            =   $   EXPORTS     MTR_SWIM;
            =   $   EXPORTS     MTR_DCRT;
            =   $   EXPORTS     MTR_DKBD;
            =   $   EXPORTS     MTR_SCRT;
            =   $   EXPORTS     MTR_SKBD;
            =   $   EXPORTS     MTR_TTY_INTR;
            =   $   EXPORTS     MTR_TTY_POLL;
            =   $   EXPORTS     MTR_IRET)
            =   $LARGE( VIDEOA EXPORTS  MTR_DCI;
            =   $   EXPORTS     MTR_DFC;
            =   $   EXPORTS     MTR_EDC;
            =   $   EXPORTS     MTR_FONT;
            =   $   EXPORTS     MTR_MDC;
            =   $   EXPORTS     MTR_MDL;
            =   $   EXPORTS     MTR_PROMPT;
            =   $   EXPORTS     MTR_RDC;
            =   $   EXPORTS     MTR_UIES;
            =   $   EXPORTS     MTR_XCA)
            =   $LARGE( FONT   EXPORTS  MTR_FONT)
            =   $LARGE( ASTLIB  EXPORTS VIDEO_INTR_A)
            =   $RESET(EXTENDED_MONITOR)
                $       NOINTVECTOR
                $SUBTITLE('External Definitions')
    1           COMMND:
                DO;

                $       INCLUDE (':F2:LEXCAL.H')
            =   $SAVE
            =   $NOLIST

                $       INCLUDE (':F2:ASCII.H')
            =   $SAVE
            =   $NOLIST
                $       INCLUDE (':F2:ASTLIB.H')
            =   $SAVE
            =   $NOLIST
                $       INCLUDE (':F2:CRTC.H')
            =   $SAVE
            =   $NOLIST
```

```
                $       INCLUDE (':F2:SUBLIB.H')
          =     $SAVE
          =     $NOLIST
                $       INCLUDE (':F2:VIDEO.H')
          =     $SAVE
          =     $NOLIST



                $       IF EXTENDED_MONITOR

                DC      BUFFER          STRUCTURE (
                            IN_POINTER          BYTE,
                            OUT_POINTER         BYTE,
                            COUNT               BYTE,
                            CHAR(100)           BYTE);

                $       ENDIF



333   1       D_KBD:          PROCEDURE(c)            EXTERNAL;
334   2         DECLARE       c                   BYTE;
335   2         END;

336   1       MTR_SWIM:               PROCEDURE       EXTERNAL INTERRUPT INT_SSTEP ;
337   2         END;

                $       IF EXTENDED_MONITOR
                P8085:                  PROCEDURE(AF,BC,DE,HL,PC,SP,PAGE)       EXTERNAL;
                  DECLARE       AF      WORD,
                                BC      WORD,
                                DE      WORD,
                                HL      WORD,
                                PC      WORD,
                                SP      WORD,
                                PAGE    WORD;
                  END;
                $       ENDIF

338   1       BOOT_207_5:     PROCEDURE               EXTERNAL;
339   2         END;

340   1       BOOT_207_8:     PROCEDURE               EXTERNAL;
341   2         END;

                $EJECT
```

```
                $       IF EXTENDED_MONITOR

                DECLARE CMD(*)  BYTE DATA(
                        'Boot',0,
                        'Color Bar',0,
                        'Continue',0,
                        'Diagnose',0,
                        'Dump',0,
                        'Examine',0,
                        'Execute',0,
                        'Fill',0,
                        'Input',0,
                        'Move',0,
                        'Output',0,
                        'Register',0,
                        'Step',0,
                        'Swap',0,
                        'Terminal',0,
                        'Trace',0,
                        'Version',0,
                        0,0);

                $       ELSE

344   1         DC      CMD_STR(4)      STRUCTURE (
                       CHAR            BYTE,           /* Identifying Character        */
                       STRNG           POINTER)        /* Address of Command String    */
                      DATA (
                       'B',@CS_BOOT,                   /* Boot String                  */
                       'V',@CS_VERS);                  /* Version String               */

345   1         DC      CS_BOOT(*)      BYTE DATA       ('Boo','t'+EOS),
                      CS_VERS(*)      BYTE DATA       ('Version 1.','2'+EOS);

                $       ENDIF


346   1         DC      (ADDR_1,ADDR_2,ADDR_3)          ADDR;
347   1         DC      BABORT          BOOL,           /* True if Boot Aborted         */
                      BYTE_1          BYTE,
                      C               BYTE,
                      CMD_IDX         BYTE,
                      CMD_PTR         BYTE,
                      DELIM           BYTE,
                      I               WORD,
                      INP_PTR         BYTE,
                      J               WORD,
                      K               WORD,
                      LINE(80)        BYTE,
                      MATCH           BYTE,
                      PTR$P           POINTER,
                      PTR_B           BASED PTR$P     BYTE,
                      SELEC           SELECTOR,
                      STALLED         BOOL,           /* I/O is currently stalled     */
                      STEP_COUNT      WORD
```

```
$       IF EXTENDED_MONITOR

COMMAND:        PROCEDURE               PUBLIC;
  DECLARE       C               BYTE;
  DO;
  CALL DPT???;          /* Display Prompt       */

  MATCH,INP_PTR,CMD_PTR,CMD_IDX=0;
  DO WHILE  CMD(CMD_PTR) <> 0  ;
    IF MATCH=0 THEN
      C = MCU(READC);
    MATCH=0;
    IF       (C=DEL OR C=BS) AND INP_PTR=0  THEN
             CALL WRITEC(BEL);
    ELSE IF  C=DEL OR C=BS                  THEN
             DO;
             CALL RUBOUT;
             INP_PTR=INP_PTR-1;
             TMP_IDX,TMP_PTR=0;
             CALL FNM;
             CMD_PTR=TMP_PTR+INP_PTR;
             CMD_IDX=TMP_IDX;
             END;
    ELSE IF  C = MCU(CMD(CMD_PTR))          THEN
             DO;
             CALL WRITEC(CMD(CMD_PTR));
             LINE(INP_PTR)=C;
             TMP_PTR,CMD_PTR=CMD_PTR+1;
             TMP_INP,INP_PTR=INP_PTR+1;
             TMP_IDX=CMD_IDX;
             CALL ANC;
             CALL FNM;
             BYTE_1=MCU(CMD(CMD_PTR));
             DO WHILE  BYTE_1<>0 AND BYTE_1=MCU(CMD(TMP_PTR+TMP_INP))  ;
               CALL ANC;
               CALL FNM;
               END;
             IF  CMD(TMP_PTR) = 0  THEN
               DO;
               MATCH=1;
               C=MCU(CMD(CMD_PTR+TMP_INP));
               END;
             END;
    ELSE
             DO;
             TMP_INP=INP_PTR;
             TMP_PTR=CMD_PTR;
             TMP_IDX=CMD_IDX;
             CALL ANC;
             CALL FNM;
             DO WHILE  CMD(TMP_PTR)<>0 AND C<>MCU(CMD(TMP_PTR+TMP_INP))  ;
               CALL ANC;
```

```
                TMP_IDX=TMP_IDX+1;
                TMP_INP=0;
                END;
              END;



            FNM:            PROCEDURE;
                DO;
                TMP_INP=0;
                DO WHILE  CMD(TMP_PTR+TMP_INP)<>0 AND TMP_INP<INP_PTR  ;
                  DO WHILE  CMD(TMP_PTR+TMP_INP)<>0 AND
                            MCU(CMD(TMP_PTR+TMP_INP))=LINE(TMP_INP) AND
                            TMP_INP<INP_PTR  ;
                    TMP_INP=TMP_INP+1;
                    END;
                  IF  TMP_INP=INP_PTR OR CMD(TMP_PTR)=0
                    THEN
                      RETURN;
                    ELSE
                      DO;
                      TMP_PTR=TMP_PTR+TMP_INP;
                      CALL ANC;
                      END;
                  END;
                END;
              END;

            $       ELSE

350   1     COMMAND:        PROCEDURE               PUBLIC;
351   2         DO;
352   3         CALL PROMPT;                        /* Display Prompt           */

353   3         CMD_IDX = 2;
354   3         DO WHILE  CMD_IDX = 2  ;
355   4           C = MCU(READC);
356   4           IF        C = 'B'  THEN
357   4                     CMD_IDX = 0;
358   4           ELSE IF   C = 'V'  THEN
359   4                     CMD_IDX = 1;
360   4           ELSE
                            CALL WRITEC(BEL);
361   4           END;
362   3         CALL WRITES(CMD_STR(CMD_IDX).STRNG);
363   3         CALL WRITEC(' ');
364   3         DO CASE  CMD_IDX  ;
365   4             CALL C_BOOT;
366   4             CALL C_VERS;
367   4             END;

368   3         END;
369   2       END;

            $       ENDIF
```

```
370   1     C_BOOT:          PROCEDURE                PUBLIC;
371   2       DECLARE        BOOT     ADDRESS,        /* Address of Boot Routine      */
                             C        BYTE,           /* Temporary Input Character    */
                             PRIMARY BOOL;            /* TRUE for boot from Prim. Dev */
372   2                      DO;

373   3         ECHO:        PROCEDURE;
374   4             DO;
375   5             CALL WRITEC(C);
376   5             C = MCU(READC);
377   5             END;
378   4           END;

                  /* Default to the DIP-Switch        */
379   3         BOOT_PAR.INDEX = INPUT(IO_DIP) AND DIP_DEVICE_MASK;
380   3         IF  BOOT_PAR.INDEX >= LENGTH(BOOT_DEVICE)  THEN
381   3           DO;
382   4           CALL WRITES(@(BEL,CR,LF,'Illegal Boot Configuratio','n'+EOS));
383   4           RETURN;
384   4           END;

385   3         BABORT = FALSE;                        /* No initial boot Abort        */
386   3         BOOT_PAR.UNIT  = 0;                    /* Default is unit 0            */
387   3         BOOT_PAR.STRNG(0) = 0;                 /* Default to NULL string       */
388   3         PRIMARY = TRUE;                        /* Default to Primary Controller*/

389   3         IF  NOT RESETF  THEN
390   3           DO;
391   4           C=MCU(READC);
392   4           IF  KY_F0+1 <= C  AND  C < KY_F0+1+LENGTH(BOOT_DEVICE)  THEN
393   4             DO;
394   5             BOOT_PAR.INDEX = C - KY_F0 - 1;
395   5             CALL WRITES(@(ESC,'p','f'+EOS));
396   5             CALL WRITEC('0'+BOOT_PAR.INDEX+1);
397   5             CALL WRITES(@(ESC,'q'+EOS));
398   5             C = MCU(READC);
399   5             END;
400   4           IF  '0' <= C  AND  C<'0'+BOOT_DEVICE(BOOT_PAR.INDEX).NUM_UNITS  THEN
401   4             DO;
402   5             BOOT_PAR.UNIT = C - '0';
403   5             CALL ECHO;
404   5             END;
405   4           IF  C = 'S'  THEN
406   4             DO;
407   5             PRIMARY = FALSE;
408   5             CALL ECHO;
409   5             END;
410   4           IF  C = ':'  THEN
411   4             DO;
412   5             1=0;
413   5             CALL WRITEC(C);
414   5             C=READC;
```

$EJECT

```
$EJECT
```

```
$       IF EXTENDED_MONITOR

C_DIAG:         PROCEDURE;
    DO;
    END;
  END;

$       ENDIF


$EJECT
```

```
            CALL WRITEC(C);
          ELSE
            CALL WRITES(@(ESC,'Fi',ESC,'G'+EOL));       /* Checkerboard */
        ADDR_3.OFFS=ADDR_3.OFFS+1;
        END;
      END;
    END;
  END;

$       ENDIF


$EJECT
```

```
/*
 *      Fill    nnnn:mmmm - pppp , qq
 */

$       IF EXTENDED_MONITOR

C_FILL:         PROCEDURE;
    DO;
    DELIM=IHA('-',@ADDR_1);
    IF DELIM<>'-' THEN
      RETURN;
    CALL WRITEC(DELIM);

    DELIM=IHV(',',@ADDR_2.OFFS,2);
    IF DELIM<>',' THEN
      RETURN;
    CALL WRITEC(DELIM);

    DELIM=IHV(CR,@WORD_1,1);
    IF DELIM<>CR THEN
      RETURN;

    DO WHILE  ADDR_1.OFFS <> ADDR_2.OFFS  ;
      CALL SFB(LOW(WORD_1),ADDR_1.BASE,ADDR_1.OFFS);
      ADDR_1.OFFS=ADDR_1.OFFS+1;
      END;
    CALL SFB(LOW(WORD_1),ADDR_1.BASE,ADDR_1.OFFS);
    END;
  END;

$       ENDIF


$EJECT
```

```
/*
 *      Move    nnnn:mmmm - pppp , qqqq
 */

$       IF EXTENDED_MONITOR

C_MOVE:         PROCEDURE;
    DO;
    DELIM=IHA('-',@ADDR_1);
    IF  DELIM<>'-'  THEN
      RETURN;
    CALL WRITEC(DELIM);

    DELIM=IHV(',',@ADDR_2.OFFS,2);
    IF  DELIM<>','  THEN
      RETURN;
    CALL WRITEC(DELIM);

    DELIM=IHV(CR,@WORD_1,2);
    IF  DELIM<>CR  THEN
      RETURN;

    IF  (ADDR_1.OFFS<=ADDR_2.OFFS AND
         ADDR_1.OFFS<=WORD_1 AND WORD_1<=ADDR_2.OFFS) OR
        (ADDR_1.OFFS>=ADDR_2.OFFS AND
         (ADDR_2.OFFS<=WORD_1 OR WORD_1<=ADDR_1.OFFS))
      THEN
        DO;
        I=-1;
        WORD_1=WORD_1+ADDR_2.OFFS-ADDR_1.OFFS;
        WORD_2=ADDR_1.OFFS;
        ADDR_1.OFFS=ADDR_2.OFFS;
        ADDR_2.OFFS=WORD_2;
        END;
      ELSE
        I=1;

    DO WHILE  ADDR_1.OFFS <> ADDR_2.OFFS  ;
      CALL  SFB(GFB(ADDR_1.BASE,ADDR_1.OFFS),ADDR_1.BASE,WORD_1);
      ADDR_1.OFFS=ADDR_1.OFFS+I;
      WORD_1=WORD_1+I;
      END;
    CALL SFB(GFB(ADDR_1.BASE,ADDR_1.OFFS),ADDR_1.BASE,WORD_1);
    END;
  END;

$       ENDIF


$EJECT
```

```
$       IF EXTENDED_MONITOR

C_STEP:         PROCEDURE;
    DO;
    STEP_COUNT=0;
    DELIM=IHV(CR,@WORD_1,2);
    IF  DELIM <> CR  THEN
      RETURN;
    CALL CRLF;

    STEP_COUNT = WORD_1 ;
    CALL SSTEP;
    END;
  END;

$       ENDIF



$       IF EXTENDED_MONITOR

SSTEP:          PROCEDURE               PUBLIC;
    DO;
    WORD_1=REG.IP;
    DO WHILE  PREFIX(GFB(REG.CS,WORD_1))  ;
      WORD_1=WORD_1+1;
      END;
    IF  GFB(REG.CS,WORD_1)=08EH AND
        (GFB(REG.CS,WORD_1+1) AND 00$011$000B)=00$010$000B
      THEN
        DO;
        CALL WRITES(@('Bad to trap through mov to SS',CR,LF+EOS));
        REG.FLAGS = REG.FLAGS AND (NOT CPU_TF) ;
        END;
      ELSE
        DO;
        REG.FLAGS = REG.FLAGS OR CPU_TF ;
        CALL XEC(REG$P);
        END;
    END;
  END;



PREFIX:         PROCEDURE(op_code)              BOOL;
  DECLARE       op_code         BYTE;
    DO;
    IF       (op_code AND 111$00$111B)=001$00$110B  THEN
            RETURN(TRUE);
    ELSE IF  op_code = 0F0H                         THEN
```

```
$       IF EXTENDED_MONITOR

C_SWAP:         PROCEDURE;
    DO;
    DELIM=IHA(',',@ADDR_1);
    IF  DELIM <> ','  THEN
      RETURN;
    DELIM=IHV(CR,@WORD_1,2);
    IF  DELIM <> CR  THEN
      RETURN;

    CALL P8085(ADDR_1.BASE,WORD_1,ADDR_1.OFFS,0,0,0,0);
    END;
  END;

$       ENDIF


$EJECT
```

```
        CALL WRITES(@(' AX    BX    CX    DX    SI    DI    BP',CR,LF+EOS));
        CALL ORV(REG.AX);
        CALL ORV(REG.BX);
        CALL ORV(REG.CX);
        CALL ORV(REG.DX);
        CALL ORV(REG.SI);
        CALL ORV(REG.DI);
        CALL OHW(REG.BP);
        CALL CRLF;

        CALL CRLF;
        CALL WRITES(@(' IP    CS    SP    SS    DS    ES   Flags',CR,LF+EOS));
        CALL ORV(REG.IP);
        CALL ORV(REG.CS);
        CALL ORV(REG.SP);
        CALL ORV(REG.SS);
        CALL ORV(REG.DS);
        CALL ORV(REG.ES);
        CALL OHW(REG.FLAGS);
        CALL CRLF;
        CALL CRLF;
        END;
      END;

$       ENDIF


$EJECT
```

```
        OUTPUT(IO_SER_A+3)=INPUT(IO_SER_A+3) AND 0FEH;
    END;
  END;



ABP:            PROCEDURE(ptr)          WORD;
  DECLARE       ptr             WORD;
    DO;
    IF  ptr < 79
      THEN
        RETURN(ptr+1);
      ELSE
        RETURN(0);
    END;
  END;



RESUME:         PROCEDURE;
    DO;
    CALL T_XMTC(XON);
    STALLED = FALSE;
    END;
  END;



STALL:          PROCEDURE;
    DO;
    STALLED = TRUE;
    CALL T_XMTC(XOFF);
    END;
  END;



T_XMTC:         PROCEDURE(c);
  DECLARE       c       BYTE;
    DO;
    CALL WCHAR(IO_SER_A,c);
    END;
  END;

$       ENDIF



$EJECT
```

```
$       IF EXTENDED_MONITOR

C_XECUTE:               PROCEDURE;
    DO;
    DELIM=IHA(CR,@ADDR_1);
    IF  DELIM<>CR  THEN
      RETURN;

    REG.CS=ADDR_1.BASE;
    REG.IP=ADDR_1.OFFS;
    CALL XEC(REG$P);
    END;
  END;

$       ENDIF


464  1      END;
```

```
00C0  A02A00          MOV     AL,CMD_IDX
00C3  B105            MOV     CL,5H
00C5  F6E1            MUL     CL
00C7  89C3            MOV     BX,AX
00C9  2EC4870B00      LES     AX,CS:CMD_STR[BX+1H]
00CE  06              PUSH    ES      ; 1
00CF  50              PUSH    AX      ; 2
00D0  E80000          CALL    WRITES
                                      ; STATEMENT # 363
00D3  B020            MOV     AL,20H
00D5  50              PUSH    AX      ; 1
00D6  E80000          CALL    WRITEC
                                      ; STATEMENT # 364
00D9  8A1E2A00        MOV     BL,CMD_IDX
00DD  B700            MOV     BH,0H
00DF  D1E3            SHL     BX,1
00E1  2EFFA7E600      JMP     CS:@8[BX]
              @8:
00E6  EA00            DW      @9
00E8  EF00            DW      @10
                                      ; STATEMENT # 365
              @9:
00EA  E80700          CALL    C_BOOT
                                      ; STATEMENT # 366
00ED  5D              POP     BP
00EE  C3              RET
              @10:
00EF  E8C201          CALL    C_VERS
                                      ; STATEMENT # 369
00F2  5D              POP     BP
00F3  C3              RET
          COMMAND     ENDP
                                      ; STATEMENT # 370
          C_BOOT      PROC NEAR
00F4  55              PUSH    BP
00F5  8BEC            MOV     BP,SP
                                      ; STATEMENT # 373
          ECHO        PROC NEAR
0276  55              PUSH    BP
0277  8BEC            MOV     BP,SP
                                      ; STATEMENT # 375
0279  FF368300        PUSH    C       ; 1
027D  E80000          CALL    WRITEC
                                      ; STATEMENT # 376
0280  E80000          CALL    READC
0283  50              PUSH    AX      ; 1
0284  E80000          CALL    MCU
0287  A28300          MOV     C,AL
                                      ; STATEMENT # 378
028A  5D              POP     BP
028B  C3              RET
          ECHO        ENDP
                                      ; STATEMENT # 379
00F7  E4FF            IN      0FFH
00F9  2407            AND     AL,7H
00FB  A20000          MOV     BOOT_PAR,AL
```

```
015F  A28300            MOV     C,AL
                                        ; STATEMENT # 400
                @13:
0162  B130              MOV     CL,30H
0164  A08300            MOV     AL,C
0167  3AC8              CMP     CL,AL
0169  7721              JA      @14
016B  A00000            MOV     AL,BOOT_PAR
016E  B205              MOV     DL,5H
0170  F6E2              MUL     DL
0172  89C3              MOV     BX,AX
0174  2E8A870200        MOV     AL,CS:BOOT_DEVICE[BX+2H]
0179  02C1              ADD     AL,CL
017B  8A168300          MOV     DL,C
017F  3AD0              CMP     DL,AL
0181  7309              JNB     @14
                                        ; STATEMENT # 402
0183  2AD1              SUB     DL,CL
0185  88165200          MOV     BOOT_PAR+52H,DL
                                        ; STATEMENT # 403
0189  E8EA00            CALL    ECHO
                                        ; STATEMENT # 405
                @14:
018C  803E830053        CMP     C,53H
0191  7508              JNZ     @15
                                        ; STATEMENT # 407
0193  C606840000        MOV     PRIMARY,0H
                                        ; STATEMENT # 408
0198  E8DB00            CALL    ECHO
                                        ; STATEMENT # 410
                @15:
019B  A08300            MOV     AL,C
019E  3C3A              CMP     AL,3AH
01A0  7539              JNZ     @16
                                        ; STATEMENT # 412
01A2  C7060C000000      MOV     I,0H
                                        ; STATEMENT # 413
01A8  50                PUSH    AX      ; 1
                @6:
01A9  E80000            CALL    WRITEC
                                        ; STATEMENT # 414
01AC  E80000            CALL    READC
01AF  A28300            MOV     C,AL
                                        ; STATEMENT # 415
                @17:
01B2  A08300            MOV     AL,C
01B5  3C0D              CMP     AL,0DH
01B7  7419              JZ      @18
01B9  A10C00            MOV     AX,I
01BC  83F84E            CMP     AX,4EH
01BF  7311              JNB     @18
                                        ; STATEMENT # 416
01C1  89C3              MOV     BX,AX
01C3  8A0E8300          MOV     CL,C
01C7  888F0200          MOV     BOOT_PAR[BX+2H],CL
                                        ; STATEMENT # 417
```

```
0231  732D              JNB     @24
0233  A02700            MOV     AL,BABORT
0236  D0D8              RCR     AL,1
0238  7226              JB      @24
                                  ; STATEMENT # 438
023A  B80000            MOV     AX,0H
023D  C41E0000          LES     BX,REGP
0241  26894718          MOV     ES:REG[BX+18H],AX
0245  268907            MOV     ES:REG[BX],AX
0248  26894704          MOV     ES:REG[BX+4H],AX
                                  ; STATEMENT # 439
024C  26C747160004      MOV     ES:REG[BX+16H],400H
                                  ; STATEMENT # 440
0252  E80000            CALL    CRLF
                                  ; STATEMENT # 441
0255  C4060000          LES     AX,REGP
0259  06                PUSH    ES      ; 1
025A  50                PUSH    AX      ; 2
025B  E80000            CALL    XEC
                                  ; STATEMENT # 443
025E  5D                POP     BP
025F  C3                RET
              @24:
0260  A02700            MOV     AL,BABORT
0263  D0D8              RCR     AL,1
0265  7305              JNB     @26
                                  ; STATEMENT # 444
0267  B86500            MOV     AX,OFFSET(@@LONG$CONSTANT$0065H)
026A  EB03              JMP     @31
                                  ; STATEMENT # 445
              @26:
026C  B87200            MOV     AX,OFFSET(@@LONG$CONSTANT$0072H)
              @4:
              @7:
              @31:
026F  0E                PUSH    CS      ; 1
0270  50                PUSH    AX      ; 2
0271  E80000            CALL    WRITES
              @27:
              @25:
                                  ; STATEMENT # 447
0274  5D                POP     BP
0275  C3                RET
              C_BOOT      ENDP
                                  ; STATEMENT # 448
              CBA       PROC NEAR
028C  55                PUSH    BP
028D  8BEC              MOV     BP,SP
                                  ; STATEMENT # 451
028F  E80000            CALL    FLAGS
0292  A32400            MOV     TFLAGS,AX
                                  ; STATEMENT # 452
0295  FB                STI
                                  ; STATEMENT # 453
0296  A90002            TEST    AX,200H
0299  7501              JNZ     @28
```

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES	
303			ADDR	LITERALLY 'STRUCTURE(BASE WORD, OFFS WORD)' 346
346	0000H	4	ADDR_1	STRUCTURE
	0000H	2	BASE	WORD
	0002H	2	OFFS	WORD
346	0004H	4	ADDR_2	STRUCTURE
	0000H	2	BASE	WORD
	0002H	2	OFFS	WORD
346	0008H	4	ADDR_3	STRUCTURE
	0000H	2	BASE	WORD
	0002H	2	OFFS	WORD
347	0027H	1	BABORT	BYTE 385* 436 443 456* 457
319	0000H	1	BACK	BYTE IN PROC (SCA) PARAMETER 319
7			BEL.	LITERALLY '007H' 360 382 425 444 445
126	0000H	1	BIIL	BYTE EXTERNAL(36)
4			BOOL	LITERALLY 'BYTE' 130 135 136 138 140 273 274 287 313 343 347 371 448
371	0022H	2	BOOT	WORD IN PROC (C_BOOT) 432* 435
338	0000H		BOOT_207_5	PROCEDURE EXTERNAL(72) STACK=0000H 342
340	0000H		BOOT_207_8	PROCEDURE EXTERNAL(73) STACK=0000H 342
342	0000H	10	BOOT_DEVICE. . . .	STRUCTURE ARRAY(2) DATA 380 392
	0000H	2	PROC	WORD 432
	0002H	1	NUM_UNITS. . .	BYTE 400
	0003H	2	DADDR.	BYTE ARRAY(2) 430 431
106	0000H	83	BOOT_PAR	STRUCTURE EXTERNAL(16)
	0000H	1	INDEX.	BYTE 379* 380 394* 396 400 430 431 432
	0001H	1	PORT	BYTE 430* 431*
	0002H	80	STRNG.	BYTE ARRAY(80) 387* 415 416* 421*
	0052H	1	UNIT	BYTE 386* 402*
8			BS	LITERALLY '008H'
127	0000H	1	BSIL	BYTE EXTERNAL(37)
343	0026H	1	BSTATUS.	BYTE PUBLIC 436
27	0000H	4	BYTEPTR.	POINTER IN PROC (INCB) PARAMETER 27
347	0028H	1	BYTE_1	BYTE
22	0000H	1	C.	BYTE IN PROC (DXMTC) PARAMETER 22
347	0029H	1	C.	BYTE 355* 356 358
371	0083H	1	C.	BYTE IN PROC (C_BOOT) 375 376* 391* 392 394 398* 400 402 405 410 413 414* 415 416 418 419* 423
30	0000H	1	C.	BYTE IN PROC (SXMTC) PARAMETER 30
334	0000H	1	C.	BYTE IN PROC (D_KBD) PARAMETER 334
9			CAN.	LITERALLY '018H'
448	028CH	40	CBA.	PROCEDURE BYTE PUBLIC STACK=0004H
307	0000H	1	CHAR	BYTE IN PROC (D_CRT) PARAMETER 307
95	0000H	1	CHAR	BYTE IN PROC (WRITEC) PARAMETER 95
90	0000H	1	CHAR	BYTE IN PROC (MCU) PARAMETER 90
347	002AH	1	CMD_IDX.	BYTE 353* 354 357* 359* 362 364
347	002BH	1	CMD_PTR.	BYTE
344	000AH	20	CMD_STR.	STRUCTURE ARRAY(4) DATA
	0000H	1	CHAR	BYTE
	0001H	4	STRNG.	POINTER 362
279	0000H	1	CMND	BYTE IN PROC (WRITK) PARAMETER 279
100			CODE_SEG	LITERALLY '0FE05H'
331	0000H	1	COL.	BYTE IN PROC (XMTSC) PARAMETER 331

```
                                                 139  140  141  142  143  144  145  146  147  148  149  150
                                                 151  152  153  154  155  156  157  158  159  160  161  162
                                                 163  164  165  166  167  168  169  170  171  172  173  174
                                                 175  176  177  178  179  180  181  182  183  184  185  186
                                                 187  188  189  190  191  192  193  194  195  196  197  198
                                                 199  200  201  202  203  204  205  206  207  208  209  210
                                                 211  212  213  214  215  216  217  218  219  220  221  222
                                                 223  224  225  226  227  228  229  230  231  232  233  234
                                                 235  236  237  238  239  240  241  242  243  244  245  246
                                                 247  248  249  250  251  252  253  254  255  256  257  258
                                                 259  260  261  262  263  264  265  266  267  268  269  270
                                                 271  272  293  294  295  296  297  298  299  300  301  343
                                                 344  345  346  347  348
107  0000H     4  DCI. . . . . .   POINTER EXTERNAL(17)
 11              DEL. . . . . .   LITERALLY '07FH'        455
347  002CH     1  DELIM. . . . . .   BYTE
108  0000H     4  DFC. . . . . .   POINTER EXTERNAL(18)
 72              DIP_0. . . . . .   LITERALLY '0000$0001B'
 73              DIP_1. . . . . .   LITERALLY '0000$0010B'
 74              DIP_2. . . . . .   LITERALLY '0000$0100B'
 75              DIP_3. . . . . .   LITERALLY '0000$1000B'
 76              DIP_4. . . . . .   LITERALLY '0001$0000B'
 77              DIP_5. . . . . .   LITERALLY '0010$0000B'
 80              DIP_50HZ . . . . .   LITERALLY '1$000$0$000B'
 78              DIP_6. . . . . .   LITERALLY '0100$0000B'
 79              DIP_7. . . . . .   LITERALLY '1000$0000B'
 84              DIP_AUTO_BOOT. . .   LITERALLY '0$000$1$000B'
 85              DIP_DEVICE_MASK. .   LITERALLY '0$000$0$111B'            379
 81              DIP_RSV_1. . . . .   LITERALLY '0$100$0$000B'
 82              DIP_RSV_2. . . . .   LITERALLY '0$010$0$000B'
 83              DIP_RSV_3. . . . .   LITERALLY '0$001$0$000B'
123  0000H   256  DMAP . . . . . .   BYTE ARRAY(256) EXTERNAL(33)
105  0000H     2  DS_SIZE. . . . . .   WORD EXTERNAL(15)
 21  0000H        DXMTC. . . . . .   PROCEDURE EXTERNAL(0) STACK=0000H
306  0000H        D_CRT. . . . . .   PROCEDURE EXTERNAL(59) STACK=0000H
333  0000H        D_KBD. . . . . .   PROCEDURE EXTERNAL(70) STACK=0000H
309  0000H        D_TTY_RES. . . . .   PROCEDURE EXTERNAL(60) STACK=0000H
109  0000H     4  D_XMTC . . . . . .   POINTER EXTERNAL(19)
373  0276H    22  ECHO . . . . . .   PROCEDURE IN PROC (C_BOOT) STACK=0006H          403  408
110  0000H     4  EDC. . . . . .   POINTER EXTERNAL(20)
111  0000H     4  EMEC . . . . . .   POINTER EXTERNAL(21)
315  0000H        ENABLE_LINES . . .   PROCEDURE EXTERNAL(63) STACK=0000H
 12              EOL. . . . . .   LITERALLY '080H'
 13              EOS. . . . . .   LITERALLY '080H'        345  382  395  397  425  444  445
 14              ESC. . . . . .   LITERALLY '01BH'        395  397
137  0000H    10  ESCP . . . . . .   STRUCTURE EXTERNAL(44)
     0000H     1      COMMAND. . . .   BYTE
     0001H     2      FUNCTION . . .   WORD
     0003H     1      MODE . . . . .   BYTE
     0004H     1      OPER_COUNT . .   BYTE
     0005H     1      OPER_INDEX . .   BYTE
     0006H     4      OPERAND. . . .   BYTE ARRAY(4)
147              F86_AF . . . . . .   LITERALLY '0000$0001$0000B'
149              F86_CF . . . . . .   LITERALLY '0000$0000$0001B'
142              F86_DF . . . . . .   LITERALLY '0100$0000$0000B'
143              F86_IF . . . . . .   LITERALLY '0010$0000$0000B'        453
```

```
204                 ICW3_S6. . . . . LITERALLY '0100$0000B'
203                 ICW3_S7. . . . . LITERALLY '1000$0000B'
215                 ICW4_86. . . . . LITERALLY '0000$0001B'
214                 ICW4_AEOI. . . . LITERALLY '0000$0010B'
212                 ICW4_BUF . . . . LITERALLY '0000$1000B'
211                 ICW4_FNM . . . . LITERALLY '0001$0000B'
213                 ICW4_MS. . . . . LITERALLY '0000$0100B'
190                 IL_KV. . . . . LITERALLY '6'
184                 IL_PERR. . . . . LITERALLY '0'
185                 IL_PSWAP . . . . LITERALLY '1'
191                 IL_PTR . . . . . LITERALLY '7'
187                 IL_S100. . . . . LITERALLY '3'
188                 IL_SER_A . . . . LITERALLY '4'
189                 IL_SER_B . . . . LITERALLY '5'
186                 IL_TIMER . . . . LITERALLY '2'
 26  0000H          INCB . . . . . PROCEDURE EXTERNAL(2) STACK=0000H
 37  0000H          INIT_ITV . . . . PROCEDURE EXTERNAL(6) STACK=0000H
                    INPUT. . . . . BUILTIN          379
347  002DH      1   INP_PTR. . . . . BYTE
150                 INT_DBZ. . . . . LITERALLY '0'
154                 INT_IOF. . . . . LITERALLY '4'
152                 INT_NMI. . . . . LITERALLY '2'
153                 INT_OBI. . . . . LITERALLY '3'
151                 INT_SSTEP. . . . LITERALLY '1'          336
175                 IO_CRTC. . . . . LITERALLY '00DCH'
155                 IO_DIP . . . . . LITERALLY '00FFH'      379
172                 IO_GENERAL . . . LITERALLY '00E0H'
157                 IO_HIADR . . . . LITERALLY '00FDH'
166                 IO_INT_MS. . . . LITERALLY '00F2H'
167                 IO_INT_SL. . . . LITERALLY '00F0H'
165                 IO_KEYBRD. . . . LITERALLY '00F4H'
174                 IO_LTPN. . . . . LITERALLY '00DEH'
158                 IO_MEM . . . . . LITERALLY '00FCH'
171                 IO_PRINTER . . . LITERALLY '00E2H'
160                 IO_RSRV_1. . . . LITERALLY '00FAH'
162                 IO_RSRV_2. . . . LITERALLY '00F8H'
163                 IO_RSRV_3. . . . LITERALLY '00F7H'
164                 IO_RSRV_4. . . . LITERALLY '00F6H'
173                 IO_RSRV_5. . . . LITERALLY '00DFH'
177                 IO_RSRV_6. . . . LITERALLY '00C0H'
169                 IO_SER_A . . . . LITERALLY '00E8H'
168                 IO_SER_B . . . . LITERALLY '00ECH'
161                 IO_SOUND . . . . LITERALLY '00F9H'
156                 IO_SWAP. . . . . LITERALLY '00FEH'
170                 IO_TIMER . . . . LITERALLY '00E4H'
159                 IO_TSTAT . . . . LITERALLY '00FBH'
176                 IO_VIDEO . . . . LITERALLY '00D8H'
179                 IO_Z207_0. . . . LITERALLY '00B0H'      342
178                 IO_Z207_1. . . . LITERALLY '00B8H'      342
181                 IO_Z217_0. . . . LITERALLY '00A8H'
180                 IO_Z217_1. . . . LITERALLY '00ACH'
 38  0000H      1   IVI. . . . . . . BYTE IN PROC (INIT_ITV) PARAMETER      38
 38  0000H      4   IVP. . . . . . . POINTER IN PROC (INIT_ITV) PARAMETER   38
347  000EH      2   J. . . . . . . . WORD
347  0010H      2   K. . . . . . . . WORD
242                 KC_ARF . . . . . LITERALLY '02H'
```

```
                                        211  212  213  214  215  216  217  218  219  220  221  222
                                        223  224  225  226  227  228  229  230  231  232  233  234
                                        235  236  237  238  239  240  241  242  243  244  245  246
                                        247  248  249  250  251  252  253  254  255  256  257  258
                                        259  260  261  262  263  264  265  266  267  268  269  270
                                        271  272  294  295  296  297  298  299  300  301  302
 324  0000H     1  MASK . . . . . .    BYTE IN PROC (SCP1) PARAMETER      324
 347  007EH     1  MATCH. . . . . .    BYTE
  89  0000H        MCU. . . . . .      PROCEDURE BYTE EXTERNAL(9) STACK=0000H          355  376  391  398

 113  0000H     4  MDC. . . . . .      POINTER EXTERNAL(23)
 114  0000H     4  MDL. . . . . .      POINTER EXTERNAL(24)
 289  0000H        MTR_IRET . . . . .  PROCEDURE EXTERNAL(56) STACK=0000H
 102               MTR_ISSUE. . . . .  LITERALLY '54H'
 281  0000H        MTR_SCRT . . . . .  PROCEDURE EXTERNAL(52) STACK=0000H
 283  0000H        MTR_SKBD . . . . .  PROCEDURE EXTERNAL(53) STACK=0000H
 336  001BH        MTR_SWIM . . . . .  PROCEDURE EXTERNAL(71) INTERRUPT(1) STACK=0000H
 285  0000H        MTR_TTY_INTR . . .  PROCEDURE EXTERNAL(54) STACK=0000H
 287  0000H        MTR_TTY_POLL . . .  PROCEDURE BYTE EXTERNAL(55) STACK=0000H
 101               MTR_VERSION. . . .  LITERALLY '12H'
 182               M_INT0 . . . . . .  LITERALLY '16'
 192               M_ITC_0. . . . . .  LITERALLY 'IO_INT_MS+0'
 193               M_ITC_1. . . . . .  LITERALLY 'IO_INT_MS+1'
 224               OCW1_ALL . . . . .  LITERALLY '1111$1111B'
 223               OCW1_M0. . . . . .  LITERALLY '0000$0001B'
 222               OCW1_M1. . . . . .  LITERALLY '0000$0010B'
 221               OCW1_M2. . . . . .  LITERALLY '0000$0100B'
 220               OCW1_M3. . . . . .  LITERALLY '0000$1000B'
 219               OCW1_M4. . . . . .  LITERALLY '0001$0000B'
 218               OCW1_M5. . . . . .  LITERALLY '0010$0000B'
 217               OCW1_M6. . . . . .  LITERALLY '0100$0000B'
 216               OCW1_M7. . . . . .  LITERALLY '1000$0000B'
 227               OCW2_EOI . . . . .  LITERALLY '0010$0000B'
 231               OCW2_L0. . . . . .  LITERALLY '0000$0001B'
 230               OCW2_L1. . . . . .  LITERALLY '0000$0010B'
 229               OCW2_L2. . . . . .  LITERALLY '0000$0100B'
 228               OCW2_OCW2. . . . .  LITERALLY '0000$0000B'
 225               OCW2_ROT . . . . .  LITERALLY '1000$0000B'
 226               OCW2_SEOI. . . . .  LITERALLY '0100$0000B'
 232               OCW3_ESMM. . . . .  LITERALLY '0100$0000B'
 234               OCW3_OCW3. . . . .  LITERALLY '0000$1000B'
 235               OCW3_POLL. . . . .  LITERALLY '0000$0100B'
 233               OCW3_SMM . . . . .  LITERALLY '0010$0000B'
 236               OCW3_SRIS. . . . .  LITERALLY '0000$0010B'
 371  0084H     1  PRIMARY. . . . . .  BYTE IN PROC (C_BOOT)       388*  407*  429
 115  0000H     4  PROMPT . . . . . .  POINTER EXTERNAL(25)        352
 128  0000H     1  PSP. . . . . .      BYTE EXTERNAL(38)
 347  0012H     4  PTRP . . . . . .    POINTER
 347  0000H     1  PTR_B. . . . . .    BYTE BASED(PTRP)
 116  0000H     4  RDC. . . . . .      POINTER EXTERNAL(26)
  92  0000H        READC. . . . . .    PROCEDURE BYTE EXTERNAL(10) STACK=0000H          355  376  391  398
                                        414  419  455
 276  0000H        READK. . . . . .    PROCEDURE BYTE EXTERNAL(50) STACK=0000H
 293  0000H    28  REG. . . . . .      STRUCTURE BASED(REGP)
      0000H     2      ES . . . . . .  WORD       438*
      0002H     2      SS . . . . . .  WORD
```

```
  32  0000H          VIDEO_INTR_A . . .    PROCEDURE EXTERNAL(4) STACK=0000H
  44                 VID_CDC. . . . . .    LITERALLY 'IO_VIDEO+1'
  43                 VID_CDD. . . . . .    LITERALLY 'IO_VIDEO+0'
  42                 VID_CMD. . . . . .    LITERALLY 'IO_VIDEO+0'
  45                 VRMM_DAT . . . . .    LITERALLY 'IO_VIDEO+2'
  47                 VRMM_MPC . . . . .    LITERALLY 'IO_VIDEO+3'
  46                 VRMM_MPD . . . . .    LITERALLY 'IO_VIDEO+2'
  18                 VT . . . . . .        LITERALLY '00BH'
 103  0000H      5  WIP. . . . . .        BYTE ARRAY(5) EXTERNAL(13)
 349  001AH      2  WORD_1 . . . . . .    WORD
 349  001CH      2  WORD_2 . . . . . .    WORD
 349  001EH      2  WORD_3 . . . . . .    WORD
 349  0020H      2  WORD_4 . . . . . .    WORD
  94  0000H         WRITEC . . . . . .    PROCEDURE EXTERNAL(11) STACK=0000H       360  363  375  396  413
                                            418
  97  0000H         WRITES . . . . . .    PROCEDURE EXTERNAL(12) STACK=0000H       362  382  395  397  425
                                            444  445
 278  0000H         WRITK. . . . . .      PROCEDURE EXTERNAL(51) STACK=0000H
 119  0000H      4  XCA. . . . . .        POINTER EXTERNAL(29)
 133                XCOLOR_BACK. . . .    LITERALLY '00$111$000B'
 132                XCOLOR_FORE. . . .    LITERALLY '00$000$111B'
  34  0000H         XEC. . . . . .        PROCEDURE EXTERNAL(5) STACK=0000H        441
 140  0000H     10  XMT. . . . . .        STRUCTURE EXTERNAL(47)
      0000H      1      BURST. . . . .    BYTE
      0001H      2      BCOUNT . . . .    INTEGER
      0003H      2      COUNT. . . . .    WORD
      0005H      1      COL. . . . . .    BYTE
      0006H      1      ROW. . . . . .    BYTE
      0007H      1      COLOR. . . . .    BYTE
      0008H      1      GRAPHIC. . . .    BYTE
      0009H      1      REVERSE. . . .    BYTE
 330  0000H         XMTSC. . . . . .      PROCEDURE EXTERNAL(69) STACK=0000H
  19                XOFF . . . . . .      LITERALLY '013H'
  20                XON. . . . . .        LITERALLY '011H'
 131                XREVERSE . . . . .    LITERALLY '10$000$000B'


MODULE INFORMATION:

     CODE AREA SIZE     = 02B9H    697D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0085H    133D
     MAXIMUM STACK SIZE = 000AH     10D
     2272 LINES READ
     0 PROGRAM WARNINGS
     0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION
```

```
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:H19CRT.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:IODEF.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:IOLIB.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:ITCDEF.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:MISC.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:MTR100.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:P6821.H')
=    $SAVE
=    $NOLIST
     $       INCLUDE (':F2:VIDEOA.H')
=    $SAVE
=    $NOLIST

231  1       DECLARE INITIAL_VECTORS(*)      POINTER DATA (
                         @MTR_DCI,
                         @MTR_DFC,
                         @MTR_SKBD,
                         @MTR_EDC,
                         0,
                         @MTR_FONT,
                         @MTR_MDC,
                         @MTR_MDL,
                         @MTR_PROMPT,
                         @MTR_RDC,
                         @MTR_SCRT,
                         @MTR_UIES,
                         @MTR_XCA
                         );

             $SUBTITLE('Initialization Routines')
```

```
                /*
                 *      'INIT_68A21' initializes the General/Printer 68A21.
                 *
                 *      NOTE:   To provide a viable system, the interrupts
                 *              in the data port should always be enabled.
                 *              This is so that it is never necessary to
                 *              examine this port before outputting to it.
                 *              Such an output might accidentally clear a
                 *              pending interrupt.  Instead, any interrupt
                 *              enable/disable should be done in the con-
                 *              trol port of the 6821.
                 *
                 *      The various ports and pins are initialized as follows:
                 *
                 *          Port A
                 *            bits 1-0:         Output          Printer Data, bits 1-0
                 *            bit 2:            Output,1;       NOT Strobe
                 *            bit 3:            Output,1;       NOT Prime
                 *            bit 4:            Input;          Vertical Synchronization
                 *            bit 5:            Output,1;       Video Interrupt Enable
                 *            bit 6:            Input;          Light Pen Switch
                 *            bit 7:            Output,1;       Light Pen Interrupt Enable
                 *
                 *          CA1:                Low to High;    Light Pen Strobe
                 *
                 *          CA2:                Low to High;    Video Interrupt
                 *
                 *          Port B
                 *            bit 0:            Input;          Busy
                 *            bit 1:            Input;          NOT Fault
                 *            bits 7-2:         Output,0;       Printer Data, bits 7-2
                 *
                 *          CB1:                Input;          Printer Acknowledge
                 *
                 *          CB2:                Input;          Printer Busy
                 *
                 */

242   1       INIT_68A21:     PROCEDURE;
243   2           DO;

                    /*
                     *  Initialize Port A
                     */
244   3           OUTPUT(GEN_CNT)=(P6821_C2_1 OR P6821_ORA OR P6821_C1_1);
245   3           OUTPUT(GEN_DAT)=0000$1100B;         /* Initialize Data          */
246   3           OUTPUT(GEN_CNT)=(P6821_C2_1 OR P6821_C1_1);
247   3           OUTPUT(GEN_DIR)=1010$1111B;         /* Set direction            */
248   3           OUTPUT(GEN_CNT)=(P6821_C2_1 OR P6821_ORA OR P6821_C1_1);
249   3           OUTPUT(GEN_DAT)=1010$1100B;         /* Enable Interrupts        */

                    /*
```

```
/*
 *      INIT_8259_85    - Initialize 8259 for 8085
 *
 *          INIT_8259_85 initializes the 8259's for the 8085 mode
 *          of operation.
 *
 */

$       IF EXTENDED_MONITOR

INIT_8259_85:   PROCEDURE               PUBLIC;
    DO;
    OUTPUT(M_ITC_0)=ICW1_ICW1 OR ICW1_LTIM OR ICW1_IC4;
    OUTPUT(M_ITC_1)=0;
    OUTPUT(M_ITC_1)=ICW3_S3;
    OUTPUT(M_ITC_1)=ICW4_FNM OR ICW4_AEOI;
    OUTPUT(M_ITC_1)=OCW1_ALL;           /* Mask all interrupts  */
    OUTPUT(S_ITC_0)=ICW1_ICW1 OR ICW1_LTIM OR ICW1_IC4;
    OUTPUT(S_ITC_1)=0FFH;
    OUTPUT(S_ITC_1)=3;
    OUTPUT(S_ITC_1)=ICW4_FNM OR ICW4_AEOI;
    OUTPUT(S_ITC_1)=OCW1_ALL;           /* Mask all interrupts  */
    END;
  END;

$       ENDIF


$EJECT
```

```
271   1       DECLARE         INITIAL_DS      STRUCTURE (
                               WIP(5)          BYTE,
                               VERSION         BYTE,
                               DS_SIZE         WORD)
                              DATA (
                               0EAH,000H,000H,0FFH,0FFH,       /* WIP(0-4)             */
                               MTR_VERSION,                    /* BCD Monitor Version  */
                               1024                            /* Maximum Size for DS  */
                              );

272   1       INIT_DATA_SEGMENT:      PROCEDURE;
273   2           DO;
274   3           CALL MOVB(@INITIAL_DS,@WIP(0),SIZE(INITIAL_DS));
              $   IF 0=1
                  WIP(0)=0EAH;                /* Far Jump to MTR-100 Reset            */
                  WIP(1),WIP(2)=0;
                  WIP(3),WIP(4)=0FFH;
                  VERSION=MTR_VERSION;        /* BCD Monitor Version Identifier       */
                  DS_SIZE=1024;               /* Maximum Size of Data Segment         */
              $   ENDIF
275   3           BIIL=M_INT0;                /* Master     Interrupt Level           */
276   3           BSIL=S_INT0;                /* Base Slave Interrupt Level           */
277   3           FONTSIZE=0357H;             /* Font Table Size in Bytes             */
278   3           END;
279   2         END;


              $EJECT
```

```
290   1       INIT_TERMINAL:          PROCEDURE;
291   2         DECLARE       I               WORD;
292   2           DO;
                  /*
                   *  Initialize RAM Vectors to ROM Routines
                   */
293   3           CALL MOVW(@INITIAL_VECTORS,@VECTORS,SIZE(VECTORS)/2);

                  /*
                   *  Initialize H-19 Parameters
                   */
294   3           H19_PAR.CPL = 80;           /* 80 Characters Per Line              */
295   3           H19_PAR.LPS = 25;           /* 25 Lines Per Screen                 */
296   3           H19_PAR.SLI = 24;           /* 24 is Status Line Index             */
297   3           H19_PAR.DSC = 9;            /* Displayed Scan Lines Per Character  */
298   3           H19_PAR.SPC = 11;           /* 11 Scan Lines Per Character         */
299   3           XMT.BURST   = 960/60;       /* 9600 Baud, or 16 Char/Sec           */

                  /*
                   *  Initialize Keyboard and Display Maps
                   */
300   3           DO  I=0 TO LAST(DMAP)  ;
301   4             KMAP(I),DMAP(I)=I;
302   4             END;
            $     IF 0=1
                  DO  I='a'-' ' TO DEL-' '  ;
                    DMAP(I)=I-('a'-'A');
                    END;
            $     ENDIF
            $     IF 0=0
303   3           DO I=DEL+1-' ' TO (DEL+1-' ')+(DEL-'^'-1)  ;
304   4             DMAP(I)=I-(DEL+1-' ')+'^'-' ';
305   4             END;
            $     ENDIF

                  /*
                   *  Reset Terminal to Power-Up Configuration
                   */
306   3           CALL P_RES;

                  /*
                   *  Enable VSYNC Interrupts
                   */
307   3           OUTPUT(GEN_CNT)=INPUT(GEN_CNT) OR P6821_C2_0;
308   3           END;
309   2         END;



310   1       END;
```

```
006E  E6E3              OUT     0E3H
                                        ; STATEMENT # 251
0070  B000              MOV     AL,0H
0072  E6E2              OUT     0E2H
                                        ; STATEMENT # 252
0074  B012              MOV     AL,12H
0076  E6E3              OUT     0E3H
                                        ; STATEMENT # 253
0078  B0FC              MOV     AL,0FCH
007A  E6E2              OUT     0E2H
                                        ; STATEMENT # 254
007C  B016              MOV     AL,16H
007E  E6E3              OUT     0E3H
                                        ; STATEMENT # 256
0080  5D                POP     BP
0081  C3                RET
                INIT_68A21      ENDP
                                        ; STATEMENT # 257
                INIT_8259_88      PROC NEAR
0082  55                PUSH    BP
0083  8BEC              MOV     BP,SP
                                        ; STATEMENT # 259
0085  B019              MOV     AL,19H
0087  E6F2              OUT     0F2H
                                        ; STATEMENT # 260
0089  A00000            MOV     AL,BIIL
008C  B1F8              MOV     CL,0F8H
008E  22C1              AND     AL,CL
0090  E6F3              OUT     0F3H
                                        ; STATEMENT # 261
0092  B008              MOV     AL,8H
0094  E6F3              OUT     0F3H
                                        ; STATEMENT # 262
0096  B013              MOV     AL,13H
0098  E6F3              OUT     0F3H
                                        ; STATEMENT # 263
009A  B0FF              MOV     AL,0FFH
009C  E6F3              OUT     0F3H
                                        ; STATEMENT # 264
009E  B019              MOV     AL,19H
00A0  E6F0              OUT     0F0H
                                        ; STATEMENT # 265
00A2  A00000            MOV     AL,BSIL
00A5  22C1              AND     AL,CL
00A7  E6F1              OUT     0F1H
                                        ; STATEMENT # 266
00A9  B003              MOV     AL,3H
00AB  E6F1              OUT     0F1H
                                        ; STATEMENT # 267
00AD  B013              MOV     AL,13H
00AF  E6F1              OUT     0F1H
                                        ; STATEMENT # 268
00B1  B0FF              MOV     AL,0FFH
00B3  E6F1              OUT     0F1H
                                        ; STATEMENT # 270
00B5  5D                POP     BP
```

```
                                        ; STATEMENT # 286
0112  FE060200        INC     I
0116  75CE            JNZ     @1
            @2:
                                        ; STATEMENT # 287
0118  A00000          MOV     AL,BIIL
011B  0406            ADD     AL,6H
011D  50              PUSH    AX      ; 1
011E  BA0000          MOV     DX,SEG(VIDEO_INTR_A)
0121  B80000          MOV     AX,OFFSET(VIDEO_INTR_A)
0124  52              PUSH    DX      ; 2
0125  50              PUSH    AX      ; 3
0126  E80000          CALL    INIT_ITV
                                        ; STATEMENT # 289
0129  5D              POP     BP
012A  C3              RET
          INIT_IV     ENDP
                                        ; STATEMENT # 290
          INIT_TERMINAL     PROC NEAR
012B  55              PUSH    BP
012C  8BEC            MOV     BP,SP
                                        ; STATEMENT # 293
012E  BE0000          MOV     SI,OFFSET(INITIAL_VECTORS)
0131  BF0000          MOV     DI,OFFSET(VECTORS)
0134  B91A00          MOV     CX,1AH
0137  1E              PUSH    DS      ; 1
0138  07              POP     ES      ; 1
0139  0E              PUSH    CS      ; 1
013A  1F              POP     DS      ; 1
013B  FC              CLD
013C  F2A5            REP MOVSW
013E  06              PUSH    ES      ; 1
013F  1F              POP     DS      ; 1
                                        ; STATEMENT # 294
0140  C606000050      MOV     H19_PAR,50H
                                        ; STATEMENT # 295
0145  C606020019      MOV     H19_PAR+2H,19H
                                        ; STATEMENT # 296
014A  C606040018      MOV     H19_PAR+4H,18H
                                        ; STATEMENT # 297
014F  C606010009      MOV     H19_PAR+1H,9H
                                        ; STATEMENT # 298
0154  C60605000B      MOV     H19_PAR+5H,0BH
                                        ; STATEMENT # 299
0159  C606000010      MOV     XMT,10H
                                        ; STATEMENT # 300
015E  C70600000000    MOV     I,0H
            @3:
0164  A10000          MOV     AX,I
0167  3DFF00          CMP     AX,0FFH
016A  7710            JA      @4
                                        ; STATEMENT # 301
016C  89C3            MOV     BX,AX
016E  88870000        MOV     DMAP[BX],AL
0172  88870000        MOV     KMAP[BX],AL
                                        ; STATEMENT # 302
```

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
7			BEL. . . . . LITERALLY '007H'
67	0000H	1	BIIL . . . . . BYTE EXTERNAL(31) 260 275* 284 287
4			BOOL . . . . . LITERALLY 'BYTE' 71 76 77 79 81 192
47	0000H	83	BOOT_PAR . . . . . STRUCTURE EXTERNAL(11)
	0000H	1	INDEX. . . . . BYTE
	0001H	1	PORT . . . . . BYTE
	0002H	80	STRNG. . . . . BYTE ARRAY(80)
	0052H	1	UNIT . . . . . BYTE
8			BS . . . . . LITERALLY '008H'
68	0000H	1	BSIL . . . . . BYTE EXTERNAL(32) 265 276* 285
27	0000H	4	BYTEPTR. . . . . POINTER IN PROC (INCB) PARAMETER 27
30	0000H	1	C. . . . . BYTE IN PROC (SXMTC) PARAMETER 30
22	0000H	1	C. . . . . BYTE IN PROC (DXMTC) PARAMETER 22
9			CAN. . . . . LITERALLY '018H'
214	0000H	1	CHAR . . . . . BYTE IN PROC (MTR_EDC) PARAMETER 214
83	0000H	1	CHAR . . . . . BYTE IN PROC (S_CRT) PARAMETER 83
211	0000H	1	CHAR . . . . . BYTE IN PROC (MTR_DFC) PARAMETER 211
217	0000H	1	CHAR_DST . . . . . BYTE IN PROC (MTR_MDC) PARAMETER 217
217	0000H	1	CHAR_SRC . . . . . BYTE IN PROC (MTR_MDC) PARAMETER 217
41			CODE_SEG . . . . . LITERALLY '0FE05H'
86	0000H	1	COL. . . . . BYTE IN PROC (DCA) PARAMETER 86
75	0000H	7	COLOR. . . . . STRUCTURE EXTERNAL(36)
	0000H	1	FORE . . . . . BYTE
	0001H	1	BACK . . . . . BYTE
	0002H	1	MASK . . . . . BYTE
	0003H	1	CLEAR. . . . . BYTE
	0004H	1	PAINTED. . . . BYTE
	0005H	1	FONT . . . . . BYTE
	0006H	1	COMP_FONT. . . BYTE
211	0000H	2	COMPF. . . . . WORD IN PROC (MTR_DFC) PARAMETER 211
217	0000H	1	COUNT. . . . . BYTE IN PROC (MTR_MDC) PARAMETER 217
214	0000H	1	COUNT. . . . . BYTE IN PROC (MTR_EDC) PARAMETER 214
10			CR . . . . . LITERALLY '00DH'
76	0000H	3	CRTC_CURSOR. . . . STRUCTURE EXTERNAL(37)
	0000H	2	START. . . . . WORD
	0002H	1	UPDATE . . . . BYTE
77	0000H	3	CRTC_DISPLAY . . . STRUCTURE EXTERNAL(38)
	0000H	2	START. . . . . WORD
	0002H	1	UPDATE . . . . BYTE
2			DC . . . . . LITERALLY 'DECLARE' 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 197 198 199 200

```
      0008H     1       VRAM_SIZE. . .    BYTE
  225 0000H     1   HORZ . . . . . .      BYTE IN PROC (MTR_RDC) PARAMETER          225
  211 0000H     1   HORZ . . . . . .      BYTE IN PROC (MTR_DFC) PARAMETER          211
   65 0000H     1   HORZ_CHAR. . . . .    BYTE EXTERNAL(29)
   16                HT . . . . . . . .    LITERALLY '009H'
  281 0002H     1   I. . . . . . . . .    BYTE IN PROC (INIT_IV)        283*  283  284  285  286
  291 0000H     2   I. . . . . . . . .    WORD IN PROC (INIT_TERMINAL)        300*  300  301  302  303*  303
                                           304  305
  139               ICW1_A7A5. . . . .    LITERALLY '1110$0000B'
  142               ICW1_AD14. . . . .    LITERALLY '0000$0100B'
  144               ICW1_IC4 . . . . .    LITERALLY '0000$0001B'        259  264
  140               ICW1_ICW1. . . . .    LITERALLY '0001$0000B'        259  264
  141               ICW1_LTIM. . . . .    LITERALLY '0000$1000B'        259  264
  143               ICW1_SNGL. . . . .    LITERALLY '0000$0010B'
  145               ICW2_A15A8 . . . .    LITERALLY '1111$1111B'
  153               ICW3_S0. . . . . .    LITERALLY '0000$0001B'
  152               ICW3_S1. . . . . .    LITERALLY '0000$0010B'
  151               ICW3_S2. . . . . .    LITERALLY '0000$0100B'
  150               ICW3_S3. . . . . .    LITERALLY '0000$1000B'        261
  149               ICW3_S4. . . . . .    LITERALLY '0001$0000B'
  148               ICW3_S5. . . . . .    LITERALLY '0010$0000B'
  147               ICW3_S6. . . . . .    LITERALLY '0100$0000B'
  146               ICW3_S7. . . . . .    LITERALLY '1000$0000B'
  158               ICW4_86. . . . . .    LITERALLY '0000$0001B'        262  267
  157               ICW4_AEOI. . . . .    LITERALLY '0000$0010B'        262  267
  155               ICW4_BUF . . . . .    LITERALLY '0000$1000B'
  154               ICW4_FNM . . . . .    LITERALLY '0001$0000B'        262  267
  156               ICW4_MS. . . . . .    LITERALLY '0000$0100B'
  133               IL_KV. . . . . . .    LITERALLY '6'           287
  127               IL_PERR. . . . . .    LITERALLY '0'
  128               IL_PSWAP . . . . .    LITERALLY '1'
  134               IL_PTR . . . . . .    LITERALLY '7'
  130               IL_S100. . . . . .    LITERALLY '3'
  131               IL_SER_A . . . . .    LITERALLY '4'
  132               IL_SER_B . . . . .    LITERALLY '5'
  129               IL_TIMER . . . . .    LITERALLY '2'
   26 0000H         INCB . . . . . . .    PROCEDURE EXTERNAL(2) STACK=0000H
  232 003CH    21   INIT . . . . . . .    PROCEDURE PUBLIC STACK=0004H
  271 0034H     8   INITIAL_DS . . . .    STRUCTURE DATA          274
      0000H     5       WIP. . . . . .    BYTE ARRAY(5)
      0005H     1       VERSION. . . .    BYTE
      0006H     2       DS_SIZE. . . .    WORD
  231 0000H    52   INITIAL_VECTORS. .    POINTER ARRAY(13) DATA        293
      003CH         INIT_. . . . . . .    PROCEDURE STACK=0000H
  242 0051H    49   INIT_68A21 . . . .    PROCEDURE STACK=0002H         238
  257 0082H    53   INIT_8259_88 . . .    PROCEDURE PUBLIC STACK=0002H        235
  272 00B7H    39   INIT_DATA_SEGMENT.    PROCEDURE STACK=0004H         234
   37 0000H         INIT_ITV . . . . .    PROCEDURE EXTERNAL(6) STACK=0000H         284  285  287
  280 00DEH    77   INIT_IV. . . . . .    PROCEDURE STACK=000AH         236
  290 012BH   129   INIT_TERMINAL. . .    PROCEDURE STACK=0004H         239
                    INPUT. . . . . . .    BUILTIN           307
  118               IO_CRTC. . . . . .    LITERALLY '00DCH'
   98               IO_DIP . . . . . .    LITERALLY '00FFH'
  115               IO_GENERAL . . . .    LITERALLY '00E0H'       244  245  246  247  248  249  307
  100               IO_HIADR . . . . .    LITERALLY '00FDH'
  109               IO_INT_MS. . . . .    LITERALLY '00F2H'       259  260  261  262  263
```

```
 136  0000H        MTR_SCRT . . . . .   PROCEDURE EXTERNAL(50) STACK=0000H        231
 188  0000H        MTR_SKBD . . . . .   PROCEDURE EXTERNAL(51) STACK=0000H        231
 190  0000H        MTR_TTY_INTR . . .   PROCEDURE EXTERNAL(52) STACK=0000H
 192  0000H        MTR_TTY_POLL . . .   PROCEDURE BYTE EXTERNAL(53) STACK=0000H
 227  0000H        MTR_UIES . . . . .   PROCEDURE EXTERNAL(63) STACK=0000H        231
  42               MTR_VERSION. . . .   LITERALLY '12H'          271
 229  0000H        MTR_XCA. . . . . .   PROCEDURE EXTERNAL(64) STACK=0000H        231
 125               M_INT0 . . . . . .   LITERALLY '16'           275
 135               M_ITC_0. . . . . .   LITERALLY 'IO_INT_MS+0'        259
 136               M_ITC_1. . . . . .   LITERALLY 'IO_INT_MS+1'        260  261  262  263
 167               OCW1_ALL . . . . .   LITERALLY '1111$1111B'        263  268
 166               OCW1_M0. . . . . .   LITERALLY '0000$0001B'
 165               OCW1_M1. . . . . .   LITERALLY '0000$0010B'
 164               OCW1_M2. . . . . .   LITERALLY '0000$0100B'
 163               OCW1_M3. . . . . .   LITERALLY '0000$1000B'
 162               OCW1_M4. . . . . .   LITERALLY '0001$0000B'
 161               OCW1_M5. . . . . .   LITERALLY '0010$0000B'
 160               OCW1_M6. . . . . .   LITERALLY '0100$0000B'
 159               OCW1_M7. . . . . .   LITERALLY '1000$0000B'
 170               OCW2_EOI . . . . .   LITERALLY '0010$0000B'
 174               OCW2_L0. . . . . .   LITERALLY '0000$0001B'
 173               OCW2_L1. . . . . .   LITERALLY '0000$0010B'
 172               OCW2_L2. . . . . .   LITERALLY '0000$0100B'
 171               OCW2_OCW2. . . . .   LITERALLY '0000$0000B'
 168               OCW2_ROT . . . . .   LITERALLY '1000$0000B'
 169               OCW2_SEOI. . . . .   LITERALLY '0100$0000B'
 175               OCW3_ESMM. . . . .   LITERALLY '0100$0000B'
 177               OCW3_OCW3. . . . .   LITERALLY '0000$1000B'
 178               OCW3_POLL. . . . .   LITERALLY '0000$0100B'
 176               OCW3_SMM . . . . .   LITERALLY '0010$0000B'
 179               OCW3_SRIS. . . . .   LITERALLY '0000$0010B'
                   OUTPUT . . . . . .   BUILTIN          244*  245*  246*  247*  248*  249*  250*  251*  252*
                                         253*  254*  259*  260*  261*  262*  263*  264*  265*  266*  267*  268*
                                         307*
 198               P6821_A. . . . . .   LITERALLY '0'
 199               P6821_B. . . . . .   LITERALLY '2'
 207               P6821_C1_0 . . . .   LITERALLY '001H'
 206               P6821_C1_1 . . . .   LITERALLY '002H'        244  246  248  250  252  254
 204               P6821_C2_0 . . . .   LITERALLY '008H'        307
 203               P6821_C2_1 . . . .   LITERALLY '010H'        244  246  248  250  252  254
 202               P6821_C2_2 . . . .   LITERALLY '020H'
 200               P6821_IR01 . . . .   LITERALLY '080H'
 201               P6821_IR02 . . . .   LITERALLY '040H'
 205               P6821_ORA. . . . .   LITERALLY '004H'        244  248  250  254
 180               PRN_CNT. . . . . .   LITERALLY 'IO_PRINTER+1'          250  252  254
 181               PRN_DAT. . . . . .   LITERALLY 'IO_PRINTER'        251
 182               PRN_DIR. . . . . .   LITERALLY 'IO_PRINTER'        253
  56  0000H    4   PROMPT . . . . . .   POINTER EXTERNAL(20)
  69  0000H    1   PSP. . . . . . . .   BYTE EXTERNAL(33)
  90  0000H        P_ELIN . . . . . .   PROCEDURE EXTERNAL(46) STACK=0000H
  92  0000H        P_EPAG . . . . . .   PROCEDURE EXTERNAL(47) STACK=0000H
  88  0000H        P_HOM. . . . . . .   PROCEDURE EXTERNAL(45) STACK=0000H
  94  0000H        P_RES. . . . . . .   PROCEDURE EXTERNAL(48) STACK=0000H        306
  57  0000H    4   RDC. . . . . . . .   POINTER EXTERNAL(21)
  35  0000H    4   REGP . . . . . . .   POINTER IN PROC (XEC) PARAMETER          35
  70  0000H    4   REGP . . . . . . .   POINTER EXTERNAL(34)
```

```
SERIES-III PL/M-86 V2.0 COMPILATION OF MODULE FNCLIB
OBJECT MODULE PLACED IN :F2:FNCLIB.OBJ
COMPILER INVOKED BY:  PLM86.86 :F2:FNCLIB.PLM CODE PRINT(:TO:)


                $TITLE('MTR-101 Z-Machine Monitor ROM:  Function Library')
                $SUBTITLE('Compiler Controls')
                $       INCLUDE (':F2:COMCON.H')
          =     $SUBTITLE('Compiler Controls')
          =     $OPTIMIZE(3)
          =     $NOOVERFLOW
          =     $COMPACT
          =     $ROM
          =     $XREF
          =     $LARGE( MTR100 EXPORTS  MTR_MON;
          =     $    EXPORTS      MTR_MON;
          =     $    EXPORTS      MTR_SWIM;
          =     $    EXPORTS      MTR_DCRT;
          =     $    EXPORTS      MTR_DKBD;
          =     $    EXPORTS      MTR_SCRT;
          =     $    EXPORTS      MTR_SKBD;
          =     $    EXPORTS      MTR_TTY_INTR;
          =     $    EXPORTS      MTR_TTY_POLL;
          =     $    EXPORTS      MTR_IRET)
          =     $LARGE( VIDEOA EXPORTS  MTR_DCI;
          =     $    EXPORTS      MTR_DFC;
          =     $    EXPORTS      MTR_EDC;
          =     $    EXPORTS      MTR_FONT;
          =     $    EXPORTS      MTR_MDC;
          =     $    EXPORTS      MTR_MDL;
          =     $    EXPORTS      MTR_PROMPT;
          =     $    EXPORTS      MTR_RDC;
          =     $    EXPORTS      MTR_U1ES;
          =     $    EXPORTS      MTR_XCA)
          =     $LARGE( FONT   EXPORTS  MTR_FONT)
          =     $LARGE( ASTLIB  EXPORTS VIDEO_INTR_A)
          =     $RESET(EXTENDED_MONITOR)
                $SUBTITLE('External Definitions')
    1           FNCLIB:
                DO;

                $       INCLUDE (':F2:LEXCAL.H')
          =     $SAVE
          =     $NOLIST

                $       INCLUDE (':F2:IODEF.H')
          =     $SAVE
          =     $NOLIST

                $       INCLUDE (':F2:ASCII.H')
          =     $SAVE
          =     $NOLIST
                $       INCLUDE (':F2:ASTLIB.H')
          =     $SAVE
          =     $NOLIST
```

```
                /*
                 *      CRLF    - Carriage Return / Line-Feed
                 *
                 *          'CRLF' outputs a carriage-return/line-feed
                 *          sequence to the console.
                 *
                 */

183   1         CRLF:           PROCEDURE       PUBLIC;
184   2           DO;
185   3           CALL WRITEC(CR);
186   3           IF  NOT H19_MODE.AUTO_LF  THEN
187   3             CALL WRITEC(LF);
188   3           END;
189   2         END;



                /*
                 *      GFW     - Get Far Word
                 *
                 *          GFW gets the specified word based on the supplied
                 *          paragraph/segment and offset.
                 *
                 *      NOTE:   The actual routine which performs the work
                 *              is contained in 'ASTLIB', the assembly lan-
                 *              guage assist library.
                 *
                 *      Entry:  base    = paragraph/segment
                 *              offset  = Offset
                 *
                 *      Exit:   Returns WORD addressed by the base/offset pair
                 *
                 */

                $       IF 1=2

                GFW:            PROCEDURE(base,offset)          WORD PUBLIC;
                  DECLARE       base            WORD,
                                offset          WORD;
                    DO;
                    RETURN(GFB(base,offset)+SHL(GFB(base,offset+1),8));
                    END;
                  END;

                $       ENDIF



                /*
```

```
$       IF EXTENDED_MONITOR

IHC:            PROCEDURE(delim)                BYTE PUBLIC;
  DECLARE       delim           BYTE;
  DECLARE       C               BYTE;

    DO;
    C=MCU(READC);
    DO WHILE  (C<>delim) AND (C<>DEL) AND (C<>BS) AND (NOT VHC(C))  ;
      CALL WRITEC(BEL);
      C=MCU(READC);
      END;
    RETURN(C);
    END;
  END;

$       ENDIF



/*
 *      IHV     - Input Hex Value
 *
 *          IHV inputs a hex value from the console.  The
 *          value is to be terminated by the supplied de-
 *          limiter.  Other interpretable characters are
 *          DEL and BS.  If DEL or BS are hit, the routine
 *          returns immediately, otherwise, the value at
 *          the supplied address is modified according to
 *          the attribute specified. If 'size' is 1, then
 *          a byte value is assumed. If 'size' is 2, then
 *          a word value is assumed.
 *
 *      NOTE:   Delimiters which are valid hex characters
 *              should be avoided, and that at least one
 *              valid hex character is required before
 *              the delimiter is accepted.
 *
 *      Entry:  delim   = required value delimiter
 *              val$p   = value pointer
 *              size    = 1 for byte value, 2 for word value
 *
 *      Exit:   Returns the actual delimiter entered
 *              val$p modified if the terminating character
 *                was not 'delim'.
 *
 */

$       IF EXTENDED_MONITOR

IHV:            PROCEDURE(delim,val$p,size)             BYTE PUBLIC;
  DECLARE       delim           BYTE,
                val$p           POINTER,
                size            BYTE;
  DECLARE       C               BYTE,
```

```
193   3           IF  'a'<=char AND char<='z'
                    THEN
194   3             RETURN (char-'a'+'A');
195   3             ELSE
                    RETURN (char);
196   3           END;
197   2         END;



/*
 *       OHB
 *
 *           OHB outputs a hex byte to the console.
 *
 *       Entry:  dat     = byte value to output
 *
 *       Exit:   NONE
 *
 */

$       IF EXTENDED_MONITOR

OHB:              PROCEDURE(dat)          PUBLIC;
  DECLARE         dat             BYTE;

    DO;
    CALL OHC(SHR(dat,4));
    CALL OHC(dat AND 00FH);
    END;
  END;

$       ENDIF



/*
 *       OHC     - Output Hex Character
 *
 *           OHC ouputs a hex character to the console.
 *           This routine must be called carefully, as
 *           the data supplied is not checked for valid-
 *           ity.
 *
 *       Entry:  dat     = Binary value to output the hex character for
 *
 *       Exit:   NONE
 *
 */

$       IF EXTENDED_MONITOR

OHC:              PROCEDURE(dat)          PUBLIC;
  DECLARE         dat             BYTE;
```

```
                /*
                *       WRITES outputs a string of text to the console.
                *       This is accomplished by outputing the bytes spec-
                *       ified by the supplied string pointer, until a
                *       terminating byte is found.  A terminator is any
                *       byte with the MSB set.
                *
                *       WRITEC is invoked for the console output.
                *
                *   Entry:  strng$p = Pointer to terminated byte string
                *
                *   Exit:   NONE
                *
                */

213   1       WRITES:         PROCEDURE(strng$p)              PUBLIC;
214   2         DECLARE       strng$p                         POINTER;
215   2         DECLARE       I                               INTEGER,
                              STRNG   BASED   strng$p(1)      BYTE;

              $   IF 0=1
                  DO;
                  I = 0;
                  DO WHILE (STRNG(I) AND EOL)=0 ;
                    CALL WRITEC(STRNG(I));
                    I = I + 1;
                    END;
                  CALL WRITEC(STRNG(I) AND 07FH);
                  END;
              $   ELSE
216   2           DO;
217   3           I = -1 ;
218   3           WRITES1:
                    DO;
219   4             I = I+1 ;
220   4             CALL WRITEC(STRNG(I));
221   4             IF (STRNG(I) AND EOS)=0 THEN
222   4               GOTO WRITES1;
223   4             END;
224   3           END;
              $   ENDIF

225   2         END;

226   1       END;
```

```
0046  8BEC              MOV     BP,SP
                                          ; STATEMENT # 201
0048  E80000            CALL    READK
004B  A20200            MOV     T,AL
                                          ; STATEMENT # 202
004E  B1B0              MOV     CL,0B0H
0050  3AC8              CMP     CL,AL
0052  7704              JA      @5
0054  3CB9              CMP     AL,0B9H
0056  760F              JBE     @6
                @5:
0058  A00200            MOV     AL,T
005B  3C8D              CMP     AL,8DH
005D  7408              JZ      @6
005F  3CAD              CMP     AL,0ADH
0061  7404              JZ      @6
0063  3CAE              CMP     AL,0AEH
0065  7505              JNZ     @4
                @6:
                                          ; STATEMENT # 203
0067  802602007F        AND     T,7FH
                                          ; STATEMENT # 204
                @4:
006C  A00200            MOV     AL,T
006F  5D                POP     BP
0070  C3                RET
                                          ; STATEMENT # 206
                READC   ENDP
                                          ; STATEMENT # 207
                WRITEC  PROC NEAR
0071  55                PUSH    BP
0072  8BEC              MOV     BP,SP
                                          ; STATEMENT # 210
0074  FF7604            PUSH    [BP].CHAR; 1
0077  E80000            CALL    S_CRT
                                          ; STATEMENT # 212
007A  5D                POP     BP
007B  C20200            RET     2H
                WRITEC  ENDP
                                          ; STATEMENT # 213
                WRITES  PROC NEAR
007E  55                PUSH    BP
007F  8BEC              MOV     BP,SP
                                          ; STATEMENT # 217
0081  C7060000FFFF      MOV     I,0FFFFH
                WRITES1:
                                          ; STATEMENT # 219
0087  A10000            MOV     AX,I
008A  40                INC     AX
008B  A30000            MOV     I,AX
                                          ; STATEMENT # 220
008E  89C6              MOV     SI,AX
0090  C45E04            LES     BX,[BP].STRNGP
0093  26FF30            PUSH    ES:[BX].STRNG[SI]
0096  E8D8FF            CALL    WRITEC
                                          ; STATEMENT # 221
```

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
34			BEL. . . . . LITERALLY '007H'
107	0000H	1	BIIL . . . . . BYTE EXTERNAL(30)
4			BOOL . . . . . LITERALLY 'BYTE' 111 116 117 119 121 174 175
87	0000H	83	BOOT_PAR . . . . . STRUCTURE EXTERNAL(10)
	0000H	1	INDEX. . . . . BYTE
	0001H	1	PORT . . . . . BYTE
	0002H	80	STRNG. . . . . BYTE ARRAY(80)
	0052H	1	UNIT . . . . . BYTE
35			BS . . . . . LITERALLY '008H'
108	0000H	1	BSIL . . . . . BYTE EXTERNAL(31)
54	0000H	4	BYTEPTR. . . . . POINTER IN PROC (INCB) PARAMETER 54
57	0000H	1	C. . . . . BYTE IN PROC (SXMTC) PARAMETER 57
49	0000H	1	C. . . . . BYTE IN PROC (DXMTC) PARAMETER 49
36			CAN. . . . . LITERALLY '018H'
191	0004H	1	CHAR . . . . . BYTE IN PROC (MCU) PARAMETER AUTOMATIC 191 193 194 195
208	0004H	1	CHAR . . . . . BYTE IN PROC (WRITEC) PARAMETER AUTOMATIC 208 210
123	0000H	1	CHAR . . . . . BYTE IN PROC (S_CRT) PARAMETER 123
180	0000H	1	CMND . . . . . BYTE IN PROC (WRITK) PARAMETER 180
81			CODE_SEG . . . . . LITERALLY '0FE05H'
126	0000H	1	COL. . . . . BYTE IN PROC (DCA) PARAMETER 126
115	0000H	7	COLOR. . . . . STRUCTURE EXTERNAL(35)
	0000H	1	FORE . . . . . BYTE
	0001H	1	BACK . . . . . BYTE
	0002H	1	MASK . . . . . BYTE
	0003H	1	CLEAR. . . . . BYTE
	0004H	1	PAINTED. . . . BYTE
	0005H	1	FONT . . . . . BYTE
	0006H	1	COMP_FONT. . . BYTE
37			CR . . . . . LITERALLY '00DH' 185
183	0010H	24	CRLF . . . . . PROCEDURE PUBLIC STACK=0006H
116	0000H	3	CRTC_CURSOR. . . . STRUCTURE EXTERNAL(36)
	0000H	2	START. . . . . WORD
	0002H	1	UPDATE . . . . BYTE
117	0000H	3	CRTC_DISPLAY . . . STRUCTURE EXTERNAL(37)
	0000H	2	START. . . . . WORD
	0002H	1	UPDATE . . . . BYTE
2			DC . . . . . LITERALLY 'DECLARE' 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173
125	0000H		DCA. . . . . PROCEDURE EXTERNAL(43) STACK=0000H
88	0000H	4	DCI. . . . . POINTER EXTERNAL(11)
38			DEL. . . . . LITERALLY '07FH'
89	0000H	4	DFC. . . . . POINTER EXTERNAL(12)

```
        0001H     1       DSC. . . . . .    BYTE
        0002H     1       LPS. . . . . .    BYTE
        0003H     1       POVRAM . . . .    BYTE
        0004H     1       SLI. . . . . .    BYTE
        0005H     1       SPC. . . . . .    BYTE
        0006H     1       SW401. . . . .    BYTE
        0007H     1       SW402. . . . .    BYTE
        0008H     1       VRAM_SIZE. . .    BYTE
  182   0000H    16   HEX_DIGIT. . . . .    BYTE ARRAY(16) PUBLIC DATA
  105   0000H     1   HORZ_CHAR. . . . .    BYTE EXTERNAL(28)
   43               HT . . . . . . . .    LITERALLY '009H'
  215   0000H     2   I. . . . . . . . .    INTEGER IN PROC (WRITES)     217*  219*  219  220  221
   53   0000H         INCB . . . . . .    PROCEDURE EXTERNAL(2) STACK=0000H
   64   0000H         INIT_ITV . . . . .    PROCEDURE EXTERNAL(6) STACK=0000H
   27               IO_CRTC. . . . . .    LITERALLY '00DCH'
    7               IO_DIP . . . . . .    LITERALLY '00FFH'
   24               IO_GENERAL . . . .    LITERALLY '00E0H'
    9               IO_HIADR . . . . .    LITERALLY '00FDH'
   18               IO_INT_MS. . . . .    LITERALLY '00F2H'
   19               IO_INT_SL. . . . .    LITERALLY '00F0H'
   17               IO_KEYBRD. . . . .    LITERALLY '00F4H'
   26               IO_LTPN. . . . . .    LITERALLY '00DEH'
   10               IO_MEM . . . . . .    LITERALLY '00FCH'
   23               IO_PRINTER . . . .    LITERALLY '00E2H'
   12               IO_RSRV_1. . . . .    LITERALLY '00FAH'
   14               IO_RSRV_2. . . . .    LITERALLY '00F8H'
   15               IO_RSRV_3. . . . .    LITERALLY '00F7H'
   16               IO_RSRV_4. . . . .    LITERALLY '00F6H'
   25               IO_RSRV_5. . . . .    LITERALLY '00DFH'
   29               IO_RSRV_6. . . . .    LITERALLY '00C0H'
   21               IO_SER_A . . . . .    LITERALLY '00E8H'
   20               IO_SER_B . . . . .    LITERALLY '00ECH'
   13               IO_SOUND . . . . .    LITERALLY '00F9H'
    8               IO_SWAP. . . . . .    LITERALLY '00FEH'
   22               IO_TIMER . . . . .    LITERALLY '00E4H'
   11               IO_TSTAT . . . . .    LITERALLY '00FBH'
   28               IO_VIDEO . . . . .    LITERALLY '00D8H'
   31               IO_Z207_0. . . . .    LITERALLY '00B0H'
   30               IO_Z207_1. . . . .    LITERALLY '00B8H'
   33               IO_Z217_0. . . . .    LITERALLY '00A8H'
   32               IO_Z217_1. . . . .    LITERALLY '00ACH'
   65   0000H     1   IVI. . . . . . . .    BYTE IN PROC (INIT_ITV) PARAMETER          65
   65   0000H     4   IVP. . . . . . . .    POINTER IN PROC (INIT_ITV) PARAMETER       65
  143               KC_ARF . . . . . .    LITERALLY '02H'
  144               KC_ARO . . . . . .    LITERALLY '01H'
  145               KC_BEP . . . . . .    LITERALLY '07H'
  150               KC_CFI . . . . . .    LITERALLY '05H'
  151               KC_CLK . . . . . .    LITERALLY '06H'
  146               KC_DI. . . . . . .    LITERALLY '0DH'
  147               KC_EI. . . . . . .    LITERALLY '0CH'
  152               KC_KBD . . . . . .    LITERALLY '09H'
  153               KC_KBE . . . . . .    LITERALLY '08H'
  148               KC_KCF . . . . . .    LITERALLY '04H'
  149               KC_KCO . . . . . .    LITERALLY '03H'
  154               KC_MNS . . . . . .    LITERALLY '0BH'
  155               KC_MUD . . . . . .    LITERALLY '0AH'
```

```
  214  0004H     4  STRNGP . . . . . .  POINTER IN PROC (WRITES) PARAMETER AUTOMATIC      214  220  221
   56  0000H        SXMTC. . . . . .  PROCEDURE EXTERNAL(3) STACK=0000H
  122  0000H        S_CRT. . . . . .  PROCEDURE EXTERNAL(42) STACK=0000H      210
   98  0000H     4  S_XMTC . . . . . .  POINTER EXTERNAL(21)
  199  0002H     1  T. . . . . .  BYTE IN PROC (READC)      201*  202  203*  203  204
  136  0000H        TEC. . . . . .  PROCEDURE EXTERNAL(48) STACK=0000H
  175  0000H        TESTK. . . . . .  PROCEDURE BYTE EXTERNAL(50) STACK=0000H
    6                TRUE . . . . . .  LITERALLY '0FFH'
   99  0000H     4  UIES . . . . . .  POINTER EXTERNAL(22)
  101  0000H    52  VECTORS. . . . . .  POINTER ARRAY(13) EXTERNAL(24)
   85  0000H     1  VERSION. . . . . .  BYTE EXTERNAL(8)
  106  0000H     1  VERT_LINE. . . . .  BYTE EXTERNAL(29)
   59  0000H        VIDEO_INTR_A . . .  PROCEDURE EXTERNAL(4) STACK=0000H
   45                VT . . . . . .  LITERALLY '00BH'
   84  0000H     5  WIP. . . . . .  BYTE ARRAY(5) EXTERNAL(7)
  207  0071H    13  WRITEC . . . . . .  PROCEDURE PUBLIC STACK=0008H      185  187  220
  213  007EH    44  WRITES . . . . . .  PROCEDURE PUBLIC STACK=0010H
  218  0087H        WRITES1. . . . . .  LABEL IN PROC (WRITES)      222
  179  0000H        WRITK. . . . . .  PROCEDURE EXTERNAL(52) STACK=0000H
  100  0000H     4  XCA. . . . . .  POINTER EXTERNAL(23)
  114                XCOLOR_BACK. . . .  LITERALLY '00$111$000B'
  113                XCOLOR_FORE. . . .  LITERALLY '00$000$111B'
   61  0000H        XEC. . . . . .  PROCEDURE EXTERNAL(5) STACK=0000H
  121  0000H    10  XMT. . . . . .  STRUCTURE EXTERNAL(41)
       0000H     1      BURST. . . . .  BYTE
       0001H     2      BCOUNT . . . .  INTEGER
       0003H     2      COUNT. . . . .  WORD
       0005H     1      COL. . . . . .  BYTE
       0006H     1      ROW. . . . . .  BYTE
       0007H     1      COLOR. . . . .  BYTE
       0008H     1      GRAPHIC. . . .  BYTE
       0009H     1      REVERSE. . . .  BYTE
   46                XOFF . . . . . .  LITERALLY '013H'
   47                XON. . . . . .  LITERALLY '011H'
  112                XREVERSE . . . . .  LITERALLY '10$000$000B'
```

MODULE INFORMATION:

```
  CODE AREA SIZE     = 00AAH    170D
  CONSTANT AREA SIZE = 0000H      0D
  VARIABLE AREA SIZE = 0003H      3D
  MAXIMUM STACK SIZE = 0010H     16D
  1125 LINES READ
  0 PROGRAM WARNINGS
  0 PROGRAM ERRORS
```

END OF PL/M-86 COMPILATION

```
/*
 *      I2661
 *
 *          I2661 initializes the specified 2661 EPCI.
 *
 *          Entry:      base    = Base address of port
 *
 *          Exit:       NONE
 *
 */

$       IF EXTENDED_MONITOR

I2661:  PROCEDURE(base)         PUBLIC;
  DECLARE       base            WORD;

  DO;
  OUTPUT(0EAH)=04EH;
  OUTPUT(0EAH)=05DH;
  OUTPUT(0EBH)=027H;
  END;
END;

$       ENDIF



/*
 *      RCHAR
 *
 *          RCHAR returns a character from the specified EPCI.
 *          RCHAR waits for the character task-time.
 *
 *      Entry:  byte    = The base address of the EPCI
 *
 *      Exit:   Byte returned, with the parity bit stripped
 *
 */

$       IF EXTENDED MONITOR

RCHAR:  PROCEDURE(base) BYTE    PUBLIC;
  DECLARE       base            WORD;

  DO;
  DO WHILE  NOT TCHAR(base)  ;
    END;
  RETURN (INPUT(0E8H) AND 07FH);
  END;
END;
```

```
WCHAR:  PROCEDURE(base,char)    PUBLIC;
  DECLARE       base            WORD,
                char            BYTE;

  DO;
  OUTPUT(0E8H+3)=INPUT(0E8H+3) OR 1;
  DO WHILE (INPUT(0E9H) AND 001H)=0 ;
    END;
  OUTPUT(0E8H)=char;
  END;
END;

$       ENDIF


7  1    END;
```

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES			
4			BOOL	LITERALLY 'BYTE'		
2			DC	LITERALLY 'DECLARE'	3	4 5
5			FALSE.	LITERALLY '000H'		
	0000H		IOLIB.	PROCEDURE STACK=0000H		
3			LT	LITERALLY 'LITERALLY'	4	5 6
6			TRUE	LITERALLY '0FFH'		

MODULE INFORMATION:

```
     CODE AREA SIZE     = 0000H      0D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0000H      0D
     MAXIMUM STACK SIZE = 0000H      0D
     191 LINES READ
     0 PROGRAM WARNINGS
     0 PROGRAM ERRORS
```

END OF PL/M-86 COMPILATION

```
        AVAIL           BOOL
          )                     PUBLIC;

$SUBTITLE('Key-Board Procedures')
```

```
                /*
                 *      READK   - Read Key-Board
                 *
                 *      Description
                 *
                 *          READK reads data from the key-board.  This routine
                 *          waits until the key-board presents a character.
                 *
                 *      Entry:  NONE
                 *
                 *      Exit:   Returns the input character
                 *
                 */

 76   1         READK:          PROCEDURE               BYTE PUBLIC;
 77   2           DECLARE       C               BYTE;
 78   2             DO;
 79   3             DO WHILE  NOT TESTK  ;
 80   4               END;
 81   3             DISABLE;
 82   3             IF  KEY.AVAIL
                       THEN
 83   3                 DO;
 84   4                 C=KEY.CHAR;
 85   4                 KEY.AVAIL=FALSE;
 86   4                 END;
 87   3               ELSE
                         C=INPUT(KEY_DATA);
 88   3             ENABLE;
 89   3             RETURN(C);
 90   3             END;
 91   2           END;



                $EJECT
```

```
                                        ; STATEMENT # 6
                                        ; STATEMENT # 33
                                        ; STATEMENT # 69
                                        ; STATEMENT # 71
                TESTK       PROC NEAR
0000  55                    PUSH    BP
0001  8BEC                  MOV     BP,SP
                                        ; STATEMENT # 73
0003  E4F5                  IN      0F5H
0005  2401                  AND     AL,1H
0007  08C0                  OR      AL,AL
0009  B0FF                  MOV     AL,0FFH
000B  7501                  JNZ     $+3H
000D  40                    INC     AX
000E  0A060100              OR      AL,KEY+1H
0012  5D                    POP     BP
0013  C3                    RET
                                        ; STATEMENT # 75
                TESTK       ENDP
                                        ; STATEMENT # 76
                READK       PROC NEAR
0014  55                    PUSH    BP
0015  8BEC                  MOV     BP,SP
                                        ; STATEMENT # 79
                @1:
0017  E8E6FF                CALL    TESTK
001A  D0D8                  RCR     AL,1
001C  73F9                  JNB     @1
                                        ; STATEMENT # 81
001E  FA                    CLI
                                        ; STATEMENT # 82
001F  A00100                MOV     AL,KEY+1H
0022  D0D8                  RCR     AL,1
0024  730D                  JNB     @3
                                        ; STATEMENT # 84
0026  A00000                MOV     AL,KEY
0029  A20200                MOV     C,AL
                                        ; STATEMENT # 85
002C  C606010000            MOV     KEY+1H,0H
                                        ; STATEMENT # 87
0031  EB05                  JMP     @4
                @3:
0033  E4F4                  IN      0F4H
0035  A20200                MOV     C,AL
                                        ; STATEMENT # 88
                @4:
0038  FB                    STI
                                        ; STATEMENT # 89
0039  A00200                MOV     AL,C
003C  5D                    POP     BP
003D  C3                    RET
                                        ; STATEMENT # 91
                READK       ENDP
                                        ; STATEMENT # 92
                WRITK       PROC NEAR
003E  55                    PUSH    BP
```

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
4			BOOL . . . . . LITERALLY 'BYTE' 70 71
77	0002H	1	C. . . . . BYTE IN PROC (READK) 84* 87* 89
93	0004H	1	CMND . . . . . BYTE IN PROC (WRITK) PARAMETER AUTOMATIC 93 97
2			DC . . . . . LITERALLY 'DECLARE' 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69
5			FALSE. . . . . LITERALLY '000H' 85
			INPUT. . . . . BUILTIN 73 87 95
27			IO_CRTC. . . . . LITERALLY '00DCH'
7			IO_DIP . . . . . LITERALLY '00FFH'
24			IO_GENERAL . . . . LITERALLY '00E0H'
9			IO_HIADR . . . . . LITERALLY '00FDH'
18			IO_INT_MS. . . . . LITERALLY '00F2H'
19			IO_INT_SL. . . . . LITERALLY '00F0H'
17			IO_KEYBRD. . . . . LITERALLY '00F4H' 73 87 95 97
26			IO_LTPN. . . . . LITERALLY '00DEH'
10			IO_MEM . . . . . LITERALLY '00FCH'
23			IO_PRINTER . . . . LITERALLY '00E2H'
12			IO_RSRV_1. . . . . LITERALLY '00FAH'
14			IO_RSRV_2. . . . . LITERALLY '00F8H'
15			IO_RSRV_3. . . . . LITERALLY '00F7H'
16			IO_RSRV_4. . . . . LITERALLY '00F6H'
25			IO_RSRV_5. . . . . LITERALLY '00DFH'
29			IO_RSRV_6. . . . . LITERALLY '00C0H'
21			IO_SER_A . . . . . LITERALLY '00E8H'
20			IO_SER_B . . . . . LITERALLY '00ECH'
13			IO_SOUND . . . . . LITERALLY '00F9H'
8			IO_SWAP. . . . . LITERALLY '00FEH'
22			IO_TIMER . . . . . LITERALLY '00E4H'
11			IO_TSTAT . . . . . LITERALLY '00FBH'
28			IO_VIDEO . . . . . LITERALLY '00D8H'
31			IO_Z207_0. . . . . LITERALLY '00B0H'
30			IO_Z207_1. . . . . LITERALLY '00B8H'
33			IO_Z217_0. . . . . LITERALLY '00A8H'
32			IO_Z217_1. . . . . LITERALLY '00ACH'
39			KC_ARF . . . . . LITERALLY '02H'
40			KC_ARO . . . . . LITERALLY '01H'
41			KC_BEP . . . . . LITERALLY '07H'
46			KC_CFI . . . . . LITERALLY '05H'
47			KC_CLK . . . . . LITERALLY '06H'
42			KC_DI. . . . . LITERALLY '0DH'
43			KC_EI. . . . . LITERALLY '0CH'
48			KC_KBD . . . . . LITERALLY '09H'
49			KC_KBE . . . . . LITERALLY '08H'
44			KC_KCF . . . . . LITERALLY '04H'
45			KC_KCO . . . . . LITERALLY '03H'
50			KC_MNS . . . . . LITERALLY '0BH'
51			KC_MUD . . . . . LITERALLY '0AH'
52			KC_RES . . . . . LITERALLY '00H'

```
SERIES-III PL/M-86 V2.0 COMPILATION OF MODULE SUBLIB
OBJECT MODULE PLACED IN :F2:SUBLIB.OBJ
COMPILER INVOKED BY:  PLM86.86 :F2:SUBLIB.PLM CODE PRINT(:TO:)


                $TITLE('MTR-101 Z-Machine Monitor ROM:  Subroutine Library')
                $SUBTITLE('Compiler Controls')
                $       INCLUDE (':F2:COMCON.H')
= =             $SUBTITLE('Compiler Controls')
= =             $OPTIMIZE(3)
= =             $NOOVERFLOW
= =             $COMPACT
= =             $ROM
= =             $XREF
= =             $LARGE( MTR100 EXPORTS  MTR_MON;
= =             $    EXPORTS     MTR_MON;
= =             $    EXPORTS     MTR_SWIM;
= =             $    EXPORTS     MTR_DCRT;
= =             $    EXPORTS     MTR_DKBD;
= =             $    EXPORTS     MTR_SCRT;
= =             $    EXPORTS     MTR_SKBD;
= =             $    EXPORTS     MTR_TTY_INTR;
= =             $    EXPORTS     MTR_TTY_POLL;
= =             $    EXPORTS     MTR_IRET)
= =             $LARGE( VIDEOA EXPORTS  MTR_DCI;
= =             $    EXPORTS     MTR_DFC;
= =             $    EXPORTS     MTR_EDC;
= =             $    EXPORTS     MTR_FONT;
= =             $    EXPORTS     MTR_MDC;
= =             $    EXPORTS     MTR_MDL;
= =             $    EXPORTS     MTR_PROMPT;
= =             $    EXPORTS     MTR_RDC;
= =             $    EXPORTS     MTR_UIES;
= =             $    EXPORTS     MTR_XCA)
= =             $LARGE( FONT   EXPORTS  MTR_FONT)
= =             $LARGE( ASTLIB  EXPORTS VIDEO_INTR_A)
= =             $RESET(EXTENDED_MONITOR)
                $SUBTITLE('Definitions')
1               SUBLIB:
                DO;

                $       INCLUDE (':F2:LEXCAL.H')
= =             $SAVE
= =             $NOLIST

                $       INCLUDE (':F2:STRUCT.H')
= =             $SAVE


= =             /*
= =              *  STRUCT.H
= =              *
= =              *       This file defines all of the structures for
= =              *       MTR-100.
= =              *
```

```
/*
 *      IHA     - Input Hex Address
 *
 *          IHA inputs a hex address from the console.  The
 *          format enforced is:
 *
 *              nnnn:mmmm
 *
 *          where 'nnnn' is the paragraph, and 'mmmm' is the
 *          offset within the paragraph.
 *
 *      NOTE:   Any partially entered address may result
 *              in partially modified parameters.  Further-
 *              more, the same cautions about 'delim' as
 *              outlined in 'IHV()' apply since 'IHV()' is
 *              invoked.
 *
 *      Entry:  delim   = Required delimiter
 *              addr$p  = Pointer to address structure
 *
 *      Exit:   Returns actual delimiter
 *
 */

$       IF EXTENDED_MONITOR

IHA:            PROCEDURE(delim,addr$p)         BYTE PUBLIC;
  DECLARE       delim           BYTE,
                addr$p          POINTER;
  DECLARE       DELIMITER       BYTE,
                I               BYTE,
                ADDR_           BASED addr$p    ADDR;

    DO;
    I=0;
    DO WHILE  0=0  ;
      IF I=0
        THEN
          DO;
          DELIMITER=IHV(':',@ADDR_.BASE,2);
          IF DELIMITER<>':' THEN
            RETURN(DELIMITER);
          I=1;
          END;
        ELSE
          DO;
          CALL WRITEC(DELIMITER);
          DELIMITER=IHV(delim,@ADDR_.OFFS,2);
          IF DELIMITER<>delim
            THEN
              DO;
              CALL RUBOUT;
```

```
$       IF EXTENDED_MONITOR

DECLARE REG_SPEC (*) BYTE DATA
         ( 'ESSSDSDISIBPSPDXCXBXAXIPCSFIDLDHCLCHBLBHALAH' );
/*
 *      IRI     - Input Register Identifier
 *
 *          IRI inputs a valid register name, and returns the
 *          appropriate Index into the register structure.
 *
 *      Entry:  NONE
 *
 *      Exit:   Returns -1 for delete or back-space, other wise
 *              register index into structure.
 *
 */

IRI:            PROCEDURE               BYTE PUBLIC;
  DECLARE       C               BYTE,
                C1              BYTE,
                I               BYTE,
                J               BYTE,
                MATCH           BYTE,
                PTR             BYTE;
    DO;
    I = 0 ;
    DO WHILE 0=0 ;
      IF I<2 THEN
        C = MCU(READC);
      IF        I=0 AND (C=DEL OR C=BS)  THEN
              RETURN(-1);
      ELSE IF  C=DEL OR C=BS  THEN
              DO;
              CALL RUBOUT;
              I = I - 1;
              END;
      ELSE IF  I=0  THEN
              DO;
              MATCH = 0 ;
              DO J=0 TO LENGTH(REG_SPEC)-2 BY 2 ;
                IF  C = REG_SPEC(J)  THEN
                  DO;
                  MATCH = MATCH + 1 ;
                  PTR = J ;
                  END;
                END;
              IF        MATCH = 0  THEN
                      CALL WRITEC(BEL);
              ELSE IF  MATCH = 1  THEN
                      DO;
                      CALL WRITEC(C);
                      C1 = C ;
```

```
/*
 *      OHA     - Output Hex Address
 *
 *          OHA outputs the specified hex address in the same
 *          format spedified input by 'IHA()':
 *
 *              nnnn:mmmm
 *
 *          where 'nnnn' is the segment, and 'mmmm' is the
 *          offset within the segment.
 *
 *      Entry:  addr$p  = Pointer to address structure
 *
 */

$       IF EXTENDED_MONITOR

OHA:            PROCEDURE(addr$p)               PUBLIC;
  DECLARE       addr$p          POINTER;
  DECLARE       ADDR_           BASED addr$p    ADDR;
    DO;
    CALL OHW(ADDR_.BASE);
    CALL WRITEC(':');
    CALL OHW(ADDR_.OFFS);
    END;
  END;

$       ENDIF



/*
 *      RNC     - Require Next Character
 *
 *          RNC requires the next character from the input device
 *          to be the specified one.  If the character is DEL or
 *          BS, RNC returns this character as the delimiter.  If
 *          the character is any character besides the specified
 *          delimiter, RNC outputs a bell, and waits.
 *
 *      Entry:  delim   = Character required
 *
 *      Exit:   Returns the delimiter actually read as a BYTE
 *
 */

$       IF EXTENDED_MONITOR

RNC:            PROCEDURE(delim)                BYTE PUBLIC;
  DECLARE       delim           BYTE;
  DECLARE       C               BYTE;
```

```
/*
 *      RUBOUT  -- Rub out a character
 *
 *          RUBOUT rubs out a character currently displayed on
 *          the screen.  This is done by simply outputing a
 *          backspace, space, and backspace.
 *
 *      Entry:  NONE
 *
 *      Exit:   NONE
 *
 */

$       IF EXTENDED_MONITOR

RUBOUT:         PROCEDURE               PUBLIC;
    DO;
    CALL WRITES(@(BS,' ',BS+EOS));
    END;
  END;

$       ENDIF


36  1   END;
```

```
  DEFN  ADDR   SIZE  NAME, ATTRIBUTES, AND REFERENCES

     /                ADDR . . . . . .   LITERALLY 'STRUCTURE(  BASE          WORD,  OFFS              WORD)'
     8                BEL. . . . . .     LITERALLY '007H'
     4                BOOL . . . . . .   LITERALLY 'BYTE'
     9                BS . . . . . .     LITERALLY '008H'
    10                CAN. . . . . .     LITERALLY '018H'
    26  0000H     1   CHAR . . . . . .   BYTE IN PROC (MCU) PARAMETER        26
    31  0000H     1   CHAR . . . . . .   BYTE IN PROC (WRITEC) PARAMETER          31
    11                CR . . . . . .     LITERALLY '00DH'
    23  0000H         CRLF . . . . . .   PROCEDURE EXTERNAL(1) STACK=0000H
     2                DC . . . . . .     LITERALLY 'DECLARE'             3    4    5    7    8    9   10
                                           11   12   13   14   15   16   17   18   19   20
    12                DEL. . . . . .     LITERALLY '07FH'
    13                EOL. . . . . .     LITERALLY '080H'
    14                EOS. . . . . .     LITERALLY '080H'
    15                ESC. . . . . .     LITERALLY '01BH'
     5                FALSE. . . . . .   LITERALLY '000H'
    16                FF . . . . . .     LITERALLY '00CH'
    22  0000H         HEX_DIGIT. . . .   BYTE ARRAY(0) EXTERNAL(0) DATA
    17                HT . . . . . .     LITERALLY '009H'
    18                LF . . . . . .     LITERALLY '00AH'
     3                LT . . . . . .     LITERALLY 'LITERALLY'          4    5    6    8    9   10   11
                                           12   13   14   15   16   17   18   19   20   21
    25  0000H         MCU. . . . . .     PROCEDURE BYTE EXTERNAL(2) STACK=0000H
    28  0000H         READC. . . . . .   PROCEDURE BYTE EXTERNAL(3) STACK=0000H
    34  0000H     4   STRNGP . . . . .   POINTER IN PROC (WRITES) PARAMETER          34
        0000H         SUBLIB . . . . .   PROCEDURE STACK=0000H
     6                TRUE . . . . . .   LITERALLY '0FFH'
    19                VT . . . . . .     LITERALLY '00BH'
    30  0000H         WRITEC . . . . .   PROCEDURE EXTERNAL(4) STACK=0000H
    33  0000H         WRITES . . . . .   PROCEDURE EXTERNAL(5) STACK=0000H
    20                XOFF . . . . . .   LITERALLY '013H'
    21                XON. . . . . .     LITERALLY '011H'


MODULE INFORMATION:

     CODE AREA SIZE     = 0000H      0D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0000H      0D
     MAXIMUM STACK SIZE = 0000H      0D
     501 LINES READ
     0 PROGRAM WARNINGS
     0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION
```

```
      $        INCLUDE (':F2:COLORS.H')
=     $SAVE
=     $NOLIST
      $        INCLUDE (':F2:CRTC.H')
=     $SAVE
=     $NOLIST
      $        INCLUDE (':F2:DIPDEF.H')
=     $SAVE
=     $NOLIST
      $        INCLUDE (':F2:FNCLIB.H')
=     $SAVE
=     $NOLIST
      $        INCLUDE (':F2:GLOBAL.H')
=     $SAVE
=     $NOLIST
      $        INCLUDE (':F2:H19CRT.H')
=     $SAVE
=     $NOLIST
      $        INCLUDE (':F2:I8086.H')
=     $SAVE
=     $NOLIST
      $        INCLUDE (':F2:ITCDEF.H')
=     $SAVE
=     $NOLIST
      $        INCLUDE (':F2:KEYDEF.H')
=     $SAVE
=     $NOLIST
      $        INCLUDE (':F2:KEYLIB.H')
=     $SAVE
=     $NOLIST

      $        INCLUDE (':F2:MISC.H')
=     $SAVE
=     $NOLIST
      $        INCLUDE (':F2:MTR100.H')
=     $SAVE
=     $NOLIST
      $        INCLUDE (':F2:P6821.H')
=     $SAVE
=     $NOLIST


      $SUBTITLE('Global Declarations')
```

```
                /*
                 *      INIT_VIDEO      - Initialize Video
                 *
                 *          INIT_VIDEO initializes the video parameters.  Initially,
                 *          the CRT-C parameters are set.  Then the 6821 controlling
                 *          access to the video command bits is initialized.  The
                 *          6821's must be carefully initialized to avoid accidentally
                 *          'glitching' the screen.  Then, the Video RAM Mapping Mod-
                 *          ule is initialized.  Since it only controls access to the
                 *          pages of video RAM from the CPU, one need not take as much
                 *          care in the initialization.
                 *
                 *          If the first byte of the blue page is modifiable, then
                 *          it is assumed that all of the color planes are present,
                 *          and update from all of the planes is enabled, otherwise,
                 *          only the green plane is enabled (though the multiple-write
                 *          bits for all planes are set.
                 *
                 *
                 *      Entry:  NONE
                 *
                 *      Exit:   NONE
                 *
                 */

330   1         INIT_VIDEO:             PROCEDURE;
331   2             DO;
332   3             DECLARE     I               BYTE,
                                P1$PTR          POINTER,
                                P1              BASED P1$PTR    BYTE,
                                P2$PTR          POINTER,
                                P2              BASED P2$PTR    BYTE,
                                P3              BASED P1$PTR(1) BYTE,
                                T1              BYTE,
                                T2              BYTE;

                            /*
                             *  Initialize CRT-Controller, and sub-system
                             */
333   3             IF  (INPUT(IO_DIP) AND DIP_50HZ) <> 0
                      THEN
334   3                 P1$PTR=@CRTC_REG_50(0);
335   3                 ELSE
                          P1$PTR=@CRTC_REG_60(0);
336   3             DO I=0 TO 15;
337   4               CALL UCCR(I,P3(I));
338   4               END;
339   3             CALL I6821;
340   3             OUTPUT(VRMM_DAT)=0;

                            /*
                             *  Initialize Terminal Emulation Parameters
```

```
                /*
                 *  I6821
                 *
                 *      'I6821' initializes the both ports of the
                 *          6821 used primarily for video applications.
                 *          Port A is used for video commands, and Port
                 *          B is used for the Video RAM Mapping Module
                 *          offset.
                 *
                 */

372   1       I6821:  PROCEDURE;
373   2           DO;

                    /*  Initialize CA2 for input, and access peripheral register A      */
374   3           OUTPUT(VID_CDC)=(P6821_ORA OR P6821_C2_1 OR P6821_C2_0);
                    /*  Clear peripheral reg. to reset condition, all signals active low */
375   3           OUTPUT(VID_CMD)=0FFH;
                    /*  Initialize Data Direction Register CA2 for output, and          */
                    /*  Data Direction Register A access                                */
376   3           OUTPUT(VID_CDC)=(P6821_C2_2 OR P6821_C2_1 OR P6821_C2_0);
                    /*  Set all bits for output                                         */
377   3           OUTPUT(VID_CDD)=0FFH;
                    /*  Set access to peripheral register A                             */
378   3           OUTPUT(VID_CDC)=(P6821_C2_2 OR P6821_C2_1 OR P6821_C2_0 OR P6821_ORA);
                    /*  Set data direction for all lines as output                      */
379   3           OUTPUT(VRMM_MPC)=0;
380   3           OUTPUT(VRMM_MPD)=0FFH;
                    /*  Set access to peripheral register B and zero adder              */
381   3           OUTPUT(VRMM_MPC)=(P6821_ORA OR P6821_C2_2 OR P6821_C2_1 OR P6821_C2_0);
382   3           OUTPUT(VRMM_DAT)=0;

383   3             END;
384   2           END;



                $EJECT
```

```
402   1       TTY_INTR:       PROCEDURE       PUBLIC;
403   2         DECLARE       TEMP    BYTE,
                              TFLAGS  WORD;
404   2           DO;

405   3           TFLAGS = FLAGS;             /* Save Current Processor Flags         */
406   3           DISABLE;                    /* Disable ALL Interrupts               */

                  /*
                   *  Update the CRT-C Display Start Address
                   */
407   3           IF  CRTC_DISPLAY.UPDATE  THEN
408   3             DO;
409   4             OUTPUT(CRTC_RSEL)=CR_DSAH;
410   4             OUTPUT(CRTC_RVAL)=HIGH(CRTC_DISPLAY.START);
411   4             OUTPUT(CRTC_RSEL)=CR_DSAL;
412   4             OUTPUT(CRTC_RVAL)=LOW(CRTC_DISPLAY.START);
413   4             CRTC_DISPLAY.UPDATE=FALSE;
414   4             END;

                  /*
                   *  Update the CRT-C Cursor Start Address
                   */
415   3           IF  CRTC_CURSOR.UPDATE  THEN
416   3             DO;
417   4             OUTPUT(CRTC_RSEL)=CR_CSAH;
418   4             OUTPUT(CRTC_RVAL)=HIGH(CRTC_CURSOR.START);
419   4             OUTPUT(CRTC_RSEL)=CR_CSAL;
420   4             OUTPUT(CRTC_RVAL)=LOW(CRTC_CURSOR.START);
421   4             CRTC_CURSOR.UPDATE=FALSE;
422   4             END;

                  /*
                   *  Transmit any required characters
                   */
423   3           IF  XMT.COUNT <> 0  THEN
424   3             DO;
425   4             XMT.BCOUNT = XMT.BCOUNT + INT(XMT.BURST);
426   4             DO WHILE  XMT.BCOUNT > 0  AND  XMT.COUNT <> 0  ;
427   5               CALL XCA;
428   5               XMT.COUNT = XMT.COUNT - 1;
429   5               XMT.COL = XMT.COL + 1;
430   5               IF  XMT.COL = H19_PAR.CPL  THEN
431   5                 DO;
432   6                 XMT.COL = 0;
433   6                 XMT.ROW = XMT.ROW + 1;
434   6                 END;
435   5               END;
436   4             IF  XMT.COUNT = 0  THEN
437   4               CALL S$XMTC(CR);
438   4             END;
```

```
443   1     TTY_POLL:       PROCEDURE                BOOL PUBLIC;
444   2       RETURN(CRTC_CURSOR.UPDATE OR CRTC_DISPLAY.UPDATE);
445   2       END;



446   1     D_TTY_RES:      PROCEDURE                PUBLIC;
447   2       DECLARE       TFLAGS          WORD;
448   2         DO;
                /*
                 *  Disable VSYNC, Keyboard, and Light-Pen Interrupts
                 */
449   3         TFLAGS = FLAGS;
450   3         DISABLE;
451   3         OUTPUT(M_ITC_1)=INPUT(M_ITC_1) OR OCW1_M6;
452   3         IF  (TFLAGS AND F86_IF) <> 0  THEN
453   3           ENABLE;
                /*
                 *  Initialize Dumb Terminal Sub-system
                 */
454   3         CALL INIT_VIDEO;
455   3         CALL D_TTY_INIT;
                /*
                 *  Re-Enable VSYNC, Keyboard, and Light-Pen Interrupts
                 */
456   3         TFLAGS = FLAGS;
457   3         DISABLE;
458   3         OUTPUT(M_ITC_1) = INPUT(M_ITC_1) AND (NOT OCW1_M6);
459   3         IF  (TFLAGS AND F86_IF) <> 0  THEN
460   3           ENABLE;
461   3         END;
462   2       END;



              $       EJECT
```

```
499   1       D_KBD:          PROCEDURE(c)            PUBLIC;
500   2         DECLARE       c       BYTE;
501   2           DO;
502   3           CALL D$XMTC(KMAP(c));
503   3           END;
504   2         END;


          $EJECT
```

```
534   3          IF  (CA AND (XCOLOR_FORE OR XCOLOR_BACK)) <> XMT.COLOR  THEN
535   3            DO;
536   4            XMT.COLOR = CA AND (XCOLOR_FORE OR XCOLOR_BACK);
537   4            CALL TXMTC(ESC);
538   4            CALL TXMTC('m');
539   4            CALL TXMTC('0'+(CA AND XCOLOR_FORE));
540   4            CALL TXMTC('0'+SHR(CA AND XCOLOR_BACK,3));
541   4            END;

542   3          IF  ((DEL-' ')<C)  AND  XMT.GRAPHIC
                   THEN
543   3              CALL TXMTC((C-(DEL+1-' '))+'^');
544   3            ELSE
                     CALL TXMTC(C + ' ');
545   3          END;
546   2        END;


          $EJECT
```

```
565   1     REV_SCROLL:     PROCEDURE                PUBLIC;
566   2       DECLARE       TFLAGS          WORD;
567   2         DO;
568   3         CALL MDL(H19_PAR.SLI,H19_PAR.SLI-1);/* Move Status Line            */
569   3         OUTPUT(VRMM_DAT)=INPUT(VRMM_DAT)-5; /* Retreat Mapping Offset      */
570   3         CALL EDC(0,0,H19_PAR.CPL);          /* Clear top line              */

571   3         TFLAGS = FLAGS;
572   3         DISABLE;
573   3         CRTC_DISPLAY.START=CRTC_DISPLAY.START-80; /* Retreat Start Address  */
574   3         CRTC_DISPLAY.UPDATE=TRUE;
575   3         IF  (TFLAGS AND F86_IF) <> 0  THEN
576   3           ENABLE;
577   3         END;
578   2       END;



579   1     SCROLL:         PROCEDURE                PUBLIC;
580   2       DECLARE       TFLAGS          WORD;
581   2         DO;
582   3         IF  H19_MODE.STATUS  THEN
583   3           CALL MDL(H19_PAR.SLI,H19_PAR.SLI+1);/* Move Status Line          */
584   3         CALL EDC(H19_PAR.SLI,0,H19_PAR.CPL);/* Erase the new bottom line   */
585   3         OUTPUT(VRMM_DAT)=INPUT(VRMM_DAT)+5; /* Advance the mapping offset  */

586   3         TFLAGS = FLAGS;
587   3         DISABLE;
                /* Advance Display Start Address */
588   3         CRTC_DISPLAY.START=CRTC_DISPLAY.START+80;
589   3         CRTC_DISPLAY.UPDATE=TRUE;
590   3         IF  (TFLAGS AND F86_IF) <> 0  THEN
591   3           ENABLE;
592   3         END;
593   2       END;



          $EJECT
```

```
631   1       SCP1:           PROCEDURE(mask,valu)    PUBLIC;
632   2         DECLARE       mask              BYTE,
                              valu              BYTE;
633   2           DO;
634   3           H19_MODE.CURSOR = (H19_MODE.CURSOR AND mask) OR valu;
635   3           CALL SCP;
636   3           END;
637   2         END;


638   1       XMTSC:          PROCEDURE(row,col,count)        PUBLIC;
639   2         DECLARE       row               BYTE,
                              col               BYTE,
                              count             WORD;
640   2           DO;
641   3           XMT.ROW     = row;
642   3           XMT.COL     = col;
643   3           XMT.BCOUNT  = 0;
644   3           XMT.GRAPHIC,
                  XMT.REVERSE = FALSE;
645   3           XMT.COLOR   = BLACK*8 + WHITE;
646   3           XMT.COUNT   = count;
647   3           END;
648   2         END;


            $EJECT
```

```
657   2         DECLARE       TFLAGS          WORD;
658   2           DO;
659   3           TFLAGS = FLAGS;             /* Save Current Processor Flags      */
660   3           DISABLE;                    /* Disable all interrupts            */
661   3           OUTPUT(CRTC_RSEL)=reg;      /* Select specified register         */
662   3           OUTPUT(CRTC_RVAL)=val;      /* Set new value                     */
663   3           IF  (TFLAGS AND FS6_IF) <> 0  THEN
664   3             ENABLE;
665   3           END;
666   2         END;



667   1       END;
```

```
                                        ; STATEMENT # 341
0060  A20F00          MOV     H19_MODE+0FH,AL
                                        ; STATEMENT # 342
0063  B107            MOV     CL,7H
0065  51              PUSH    CX      ; 1
0066  50              PUSH    AX      ; 2
0067  E80705          CALL    SCA
                                        ; STATEMENT # 343
006A  FF1E0000        CALL    DCI
                                        ; STATEMENT # 344
006E  E80000          CALL    P_HOM
                                        ; STATEMENT # 345
0071  E80000          CALL    P_EPAG
                                        ; STATEMENT # 346
0074  B0F0            MOV     AL,0F0H
0076  50              PUSH    AX      ; 1
0077  B080            MOV     AL,80H
0079  50              PUSH    AX      ; 2
007A  E8A505          CALL    OVC
                                        ; STATEMENT # 347
007D  B900C0          MOV     CX,0C000H
0080  B80000          MOV     AX,0H
0083  A30000          MOV     P1PTR,AX
0086  890E0200        MOV     P1PTR+2H,CX
                                        ; STATEMENT # 348
008A  89C3            MOV     BX,AX
008C  8EC1            MOV     ES,CX
008E  268A17          MOV     DL,ES:P1[BX]
0091  88161500        MOV     T1,DL
                                        ; STATEMENT # 349
0095  06              PUSH    ES      ; 1
0096  50              PUSH    AX      ; 2
0097  E80000          CALL    INCB
                                        ; STATEMENT # 350
009A  C41E0000        LES     BX,P1PTR
009E  268A07          MOV     AL,ES:P1[BX]
00A1  3A061500        CMP     AL,T1
00A5  7507            JNZ     @5
                                        ; STATEMENT # 351
00A7  C606030001      MOV     H19_PAR+3H,1H
                                        ; STATEMENT # 352
00AC  EB0F            JMP     @6
                @5:
                                        ; STATEMENT # 353
00AE  C606030003      MOV     H19_PAR+3H,3H
                                        ; STATEMENT # 354
00B3  A01500          MOV     AL,T1
00B6  C41E0000        LES     BX,P1PTR
00BA  268807          MOV     ES:P1[BX],AL
                                        ; STATEMENT # 356
                @6:
00BD  B900E0          MOV     CX,0E000H
00C0  B80000          MOV     AX,0H
00C3  A30000          MOV     P1PTR,AX
00C6  890E0200        MOV     P1PTR+2H,CX
                                        ; STATEMENT # 357
```

```
0138  E8E704          CALL    OVC
                                        ; STATEMENT # 371
013B  5D              POP     BP
013C  C3              RET
              INIT_VIDEO      ENDP
                                        ; STATEMENT # 372
              I6821       PROC NEAR
013D  55              PUSH    BP
013E  8BEC            MOV     BP,SP
                                        ; STATEMENT # 374
0140  B01C            MOV     AL,1CH
0142  E6D9            OUT     0D9H
                                        ; STATEMENT # 375
0144  B0FF            MOV     AL,0FFH
0146  E6D8            OUT     0D8H
                                        ; STATEMENT # 376
0148  B038            MOV     AL,38H
014A  E6D9            OUT     0D9H
                                        ; STATEMENT # 377
014C  B0FF            MOV     AL,0FFH
014E  E6D8            OUT     0D8H
                                        ; STATEMENT # 378
0150  B03C            MOV     AL,3CH
0152  E6D9            OUT     0D9H
                                        ; STATEMENT # 379
0154  B000            MOV     AL,0H
0156  E6DB            OUT     0DBH
                                        ; STATEMENT # 380
0158  B0FF            MOV     AL,0FFH
015A  E6DA            OUT     0DAH
                                        ; STATEMENT # 381
015C  B03C            MOV     AL,3CH
015E  E6DB            OUT     0DBH
                                        ; STATEMENT # 382
0160  B000            MOV     AL,0H
0162  E6DA            OUT     0DAH
                                        ; STATEMENT # 384
0164  5D              POP     BP
0165  C3              RET
              I6821       ENDP
                                        ; STATEMENT # 385
              0_TTY_INIT      PROC NEAR
0166  55              PUSH    BP
0167  8BEC            MOV     BP,SP
                                        ; STATEMENT # 388
0169  B80000          MOV     AX,0H
016C  A30000          MOV     CRTC_DISPLAY,AX
016F  A30C00          MOV     H19_MODE+0CH,AX
0172  A30300          MOV     XMT+3H,AX
                                        ; STATEMENT # 389
0175  A00500          MOV     AL,H19_PAR+5H
0178  2C03            SUB     AL,3H
017A  0C40            OR      AL,40H
017C  A20600          MOV     H19_MODE+6H,AL
                                        ; STATEMENT # 390
017F  B000            MOV     AL,0H
```

```
01DE  E6DD              OUT     0DDH
                                ; STATEMENT # 411
01E0  B00D              MOV     AL,0DH
01E2  E6DC              OUT     0DCH
                                ; STATEMENT # 412
01E4  A10000            MOV     AX,CRTC_DISPLAY
01E7  E6DD              OUT     0DDH
                                ; STATEMENT # 413
01E9  C606020000        MOV     CRTC_DISPLAY+2H,0H
                                ; STATEMENT # 415
                @12:
01EE  A00200            MOV     AL,CRTC_CURSOR+2H
01F1  D0D8              RCR     AL,1
01F3  7317              JNB     @13
                                ; STATEMENT # 417
01F5  B00E              MOV     AL,0EH
01F7  E6DC              OUT     0DCH
                                ; STATEMENT # 418
01F9  A00100            MOV     AL,CRTC_CURSOR+1H
01FC  E6DD              OUT     0DDH
                                ; STATEMENT # 419
01FE  B00F              MOV     AL,0FH
0200  E6DC              OUT     0DCH
                                ; STATEMENT # 420
0202  A10000            MOV     AX,CRTC_CURSOR
0205  E6DD              OUT     0DDH
                                ; STATEMENT # 421
0207  C606020000        MOV     CRTC_CURSOR+2H,0H
                                ; STATEMENT # 423
                @13:
020C  833E030000        CMP     XMT+3H,0H
0211  7445              JZ      @14
                                ; STATEMENT # 425
0213  A00000            MOV     AL,XMT
0216  B400              MOV     AH,0H
0218  01060100          ADD     XMT+1H,AX
                                ; STATEMENT # 426
                @15:
021C  833E010000        CMP     XMT+1H,0H
0221  7E28              JLE     @16
0223  833E030000        CMP     XMT+3H,0H
0228  7421              JZ      @16
                                ; STATEMENT # 427
022A  FF1E0000          CALL    XCA
                                ; STATEMENT # 428
022E  FF0E0300          DEC     XMT+3H
                                ; STATEMENT # 429
0232  A00500            MOV     AL,XMT+5H
0235  FEC0              INC     AL
0237  A20500            MOV     XMT+5H,AL
                                ; STATEMENT # 430
023A  380600000         CMP     H19_PAR,AL
023E  75DC              JNZ     @15
                                ; STATEMENT # 432
0240  C606050000        MOV     XMT+5H,0H
                                ; STATEMENT # 433
```

```
                                                ; STATEMENT # 456
023E  E80000          CALL    FLAGS
0241  A30A00          MOV     TFLAGS,AX
                                                ; STATEMENT # 457
0244  FA              CLI
                                                ; STATEMENT # 458
0245  E4F3            IN      0F3H
0247  24BF            AND     AL,0BFH
0249  E6F3            OUT     0F3H
                                                ; STATEMENT # 459
024B  F7060A000002    TEST    TFLAGS,200H
0251  7401            JZ      @21
                                                ; STATEMENT # 460
0253  FB              STI
                @21:
                                                ; STATEMENT # 462
0254  5D              POP     BP
0255  C3              RET
                O_TTY_RES       ENDP
                                                ; STATEMENT # 463
                D_CRT       PROC NEAR
0256  55              PUSH    BP
0257  8BEC            MOV     BP,SP
                                                ; STATEMENT # 466
0259  8A4604          MOV     AL,[BP].C
025C  3C20            CMP     AL,20H
025E  7204            JB      @23
0260  3C7F            CMP     AL,7FH
0262  7556            JNZ     @22
                @23:
                                                ; STATEMENT # 468
0264  B007            MOV     AL,7H
0266  384604          CMP     [BP].C,AL
0269  7506            JNZ     @24
                                                ; STATEMENT # 469
026B  50              PUSH    AX              ; 1
026C  E80000          CALL    WRITK
                                                ; STATEMENT # 470
                @18:
026F  EB47            JMP     @17
                @24:
0271  807E0408        CMP     [BP].C,8H
0275  750D            JNZ     @26
0277  833E000000      CMP     HORZ_CHAR,0H
027C  7406            JZ      @26
                                                ; STATEMENT # 471
027E  FE0E0000        DEC     HORZ_CHAR
                                                ; STATEMENT # 472
0282  EBEB            JMP     @18
                @26:
0284  807E0409        CMP     [BP].C,9H
0288  7505            JNZ     @28
                                                ; STATEMENT # 473
028A  E88301          CALL    P_HT
                                                ; STATEMENT # 474
028D  EBE0            JMP     @18
```

```
034A  A00000          MOV     AL,H19_PAR
034D  FEC8            DEC     AL
034F  38060000        CMP     HORZ_CHAR,AL
0353  7306            JNB     @37
                                        ; STATEMENT # 492
0355  FE060000        INC     HORZ_CHAR
                                        ; STATEMENT # 493
0359  EB0A            JMP     @35
              @37:
035B  A01100          MOV     AL,H19_MODE+11H
035E  D0D8            RCR     AL,1
0360  7303            JNB     @35
                                        ; STATEMENT # 494
0362  E80000          CALL    CRLF
                                        ; STATEMENT # 496
              @35:
0365  E8C501          CALL    CURSOR
                                        ; STATEMENT # 498
0368  5D              POP     BP
0369  C20200          RET     2H
              O_CRT     ENDP
                                        ; STATEMENT # 499
              O_KBD     PROC NEAR
036C  55              PUSH    BP
036D  8BEC            MOV     BP,SP
                                        ; STATEMENT # 502
036F  8A5E04          MOV     BL,[BP].C
0372  B700            MOV     BH,0H
0374  FFB70000        PUSH    KMAP[BX]; 1
0378  E80000          CALL    DXMTC
                                        ; STATEMENT # 504
037B  5D              POP     BP
037C  C20200          RET     2H
              O_KBD     ENDP
                                        ; STATEMENT # 505
              TXMT      PROC NEAR
037F  55              PUSH    BP
0380  8BEC            MOV     BP,SP
                                        ; STATEMENT # 507
              TXMTC     PROC NEAR
044F  55              PUSH    BP
0450  8BEC            MOV     BP,SP
                                        ; STATEMENT # 510
0452  FF7604          PUSH    [BP].C  ; 1
0455  E80000          CALL    SXMTC
                                        ; STATEMENT # 511
0458  FF0E0100        DEC     XMT+1H
                                        ; STATEMENT # 513
045C  5D              POP     BP
045D  C20200          RET     2H
              TXMTC     ENDP
                                        ; STATEMENT # 515
0382  B80C00          MOV     AX,OFFSET(XC)
0385  1E              PUSH    DS      ; 1
0386  50              PUSH    AX      ; 2
0387  FF360600        PUSH    XMT+6H  ; 3
```

```
03ED  EB02            JMP     @27
                                        ; STATEMENT # 532
                @44:
03EF  B071            MOV     AL,71H
                @27:
03F1  50              PUSH    AX      ; 1
03F2  E85A00          CALL    TXMTC
                @45:
                                        ; STATEMENT # 534
                @43:
03F5  A01A00          MOV     AL,CA
03F8  B13F            MOV     CL,3FH
03FA  22C1            AND     AL,CL
03FC  3A060700        CMP     AL,XMT+7H
0400  7429            JZ      @46
                                        ; STATEMENT # 536
0402  A20700          MOV     XMT+7H,AL
                                        ; STATEMENT # 537
0405  B01B            MOV     AL,1BH
0407  50              PUSH    AX      ; 1
0408  E84400          CALL    TXMTC
                                        ; STATEMENT # 538
040B  B06D            MOV     AL,6DH
040D  50              PUSH    AX      ; 1
040E  E83E00          CALL    TXMTC
                                        ; STATEMENT # 539
0411  A01A00          MOV     AL,CA
0414  2407            AND     AL,7H
0416  0430            ADD     AL,30H
0418  50              PUSH    AX      ; 1
0419  E83300          CALL    TXMTC
                                        ; STATEMENT # 540
041C  A01A00          MOV     AL,CA
041F  2438            AND     AL,38H
0421  B103            MOV     CL,3H
0423  D2E8            SHR     AL,CL
0425  0430            ADD     AL,30H
0427  50              PUSH    AX      ; 1
0428  E82400          CALL    TXMTC
                                        ; STATEMENT # 542
                @46:
042B  B15F            MOV     CL,5FH
042D  A01900          MOV     AL,C
0430  3AC8            CMP     CL,AL
0432  7310            JNB     @47
0434  A00800          MOV     AL,XMT+8H
0437  D0D8            RCR     AL,1
0439  7309            JNB     @47
                                        ; STATEMENT # 543
043B  A01900          MOV     AL,C
043E  2C60            SUB     AL,60H
0440  045E            ADD     AL,5EH
0442  EB05            JMP     @29
                                        ; STATEMENT # 544
                @47:
0444  A01900          MOV     AL,C
```

```
04A6  7503              JNZ     @53
                                                ; STATEMENT # 562
04A8  E83D00            CALL    SCROLL
                @53:
                                                ; STATEMENT # 564
04AB  5D                POP     BP
04AC  C3                RET
                P_LF    ENDP
                                                ; STATEMENT # 565
                REV_SCROLL      PROC NEAR
04AD  55                PUSH    BP
04AE  8BEC              MOV     BP,SP
                                                ; STATEMENT # 568
04B0  A00400            MOV     AL,H19_PAR+4H
04B3  50                PUSH    AX      ; 1
04B4  FEC8              DEC     AL
04B6  B400              MOV     AH,0H
04B8  50                PUSH    AX      ; 2
04B9  FF1E0000          CALL    MDL
                                                ; STATEMENT # 569
04BD  E4DA              IN      0DAH
04BF  2C05              SUB     AL,5H
04C1  E6DA              OUT     0DAH
                                                ; STATEMENT # 570
04C3  B000              MOV     AL,0H
04C5  50                PUSH    AX      ; 1
04C6  50                PUSH    AX      ; 2
04C7  FF360000          PUSH    H19_PAR ; 3
04CB  FF1E0000          CALL    EDC
                                                ; STATEMENT # 571
04CF  E80000            CALL    FLAGS
04D2  A30E00            MOV     TFLAGS,AX
                                                ; STATEMENT # 572
04D5  FA                CLI
                                                ; STATEMENT # 573
04D6  832E000050        SUB     CRTC_DISPLAY,50H
                                                ; STATEMENT # 574
04DB  C6060200FF        MOV     CRTC_DISPLAY+2H,0FFH
                                                ; STATEMENT # 575
04E0  A90002            TEST    AX,200H
04E3  7401              JZ      @55
                                                ; STATEMENT # 576
04E5  FB                STI
                @55:
                                                ; STATEMENT # 578
04E6  5D                POP     BP
04E7  C3                RET
                REV_SCROLL      ENDP
                                                ; STATEMENT # 579
                SCROLL      PROC NEAR
04E8  55                PUSH    BP
04E9  8BEC              MOV     BP,SP
                                                ; STATEMENT # 582
04EB  A00F00            MOV     AL,H19_MODE+0FH
04EE  D0D8              RCR     AL,1
04F0  730D              JNB     @56
```

```
054E  A30000            MOV     CRTC_CURSOR,AX
                                ; STATEMENT # 600
0551  C6060200FF        MOV     CRTC_CURSOR+2H,0FFH
                                ; STATEMENT # 601
0556  F706120000002     TEST    TFLAGS,200H
055C  7401              JZ      @58
                                ; STATEMENT # 602
055E  FB                STI
                @58:
                                ; STATEMENT # 604
055F  5D                POP     BP
0560  C3                RET
              CURSOR    ENDP
                                ; STATEMENT # 605
              ENABLE_LINES      PROC NEAR
0561  55                PUSH    BP
0562  8BEC              MOV     BP,SP
                                ; STATEMENT # 608
0564  B006              MOV     AL,6H
0566  50                PUSH    AX      ; 1
0567  FF7604            PUSH    [BP].LINES; 2
056A  E8D100            CALL    UCCR
                                ; STATEMENT # 610
056D  5D                POP     BP
056E  C20200            RET     2H
              ENABLE_LINES      ENDP
                                ; STATEMENT # 611
              SCA       PROC NEAR
0571  55                PUSH    BP
0572  8BEC              MOV     BP,SP
                                ; STATEMENT # 614
0574  8A4606            MOV     AL,[BP].FORE
0577  A20000            MOV     COLOR,AL
                                ; STATEMENT # 615
057A  8A4604            MOV     AL,[BP].BACK
057D  A20100            MOV     COLOR+1H,AL
                                ; STATEMENT # 616
0580  B103              MOV     CL,3H
0582  D2E0              SHL     AL,CL
0584  8A4E06            MOV     CL,[BP].FORE
0587  0AC1              OR      AL,CL
0589  A20200            MOV     COLOR+2H,AL
                                ; STATEMENT # 617
058C  B207              MOV     DL,7H
058E  3AC2              CMP     AL,DL
0590  B0FF              MOV     AL,0FFH
0592  7401              JZ      $+3H
0594  40                INC     AX
0595  20060500          AND     H19_MODE+5H,AL
                                ; STATEMENT # 618
0599  8A4604            MOV     AL,[BP].BACK
059C  0AC1              OR      AL,CL
059E  32C2              XOR     AL,DL
05A0  A20300            MOV     COLOR+3H,AL
                                ; STATEMENT # 619
05A3  8A4604            MOV     AL,[BP].BACK
```

```
                XMTSC       PROC NEAR
05F8  55                    PUSH    BP
05F9  8BEC                  MOV     BP,SP
                                    ; STATEMENT # 641
05FB  8A4608                MOV     AL,[BP].ROW
05FE  A20600                MOV     XMT+6H,AL
                                    ; STATEMENT # 642
0601  8A4606                MOV     AL,[BP].COL
0604  A20500                MOV     XMT+5H,AL
                                    ; STATEMENT # 643
0607  B80000                MOV     AX,0H
060A  A30100                MOV     XMT+1H,AX
                                    ; STATEMENT # 644
060D  A20900                MOV     XMT+9H,AL
0610  A20800                MOV     XMT+8H,AL
                                    ; STATEMENT # 645
0613  C606070007            MOV     XMT+7H,7H
                                    ; STATEMENT # 646
0618  8B4604                MOV     AX,[BP].COUNT
061B  A30300                MOV     XMT+3H,AX
                                    ; STATEMENT # 648
061E  5D                    POP     BP
061F  C20600                RET     6H
                XMTSC       ENDP
                                    ; STATEMENT # 649
                OVC         PROC NEAR
0622  55                    PUSH    BP
0623  8BEC                  MOV     BP,SP
                                    ; STATEMENT # 652
0625  E4D8                  IN      0D8H
0627  8A4E06                MOV     CL,[BP].MASK
062A  F6D1                  NOT     CL
062C  22C1                  AND     AL,CL
062E  8A4E04                MOV     CL,[BP].VALU
0631  F6D1                  NOT     CL
0633  224E06                AND     CL,[BP].MASK
0636  0AC1                  OR      AL,CL
0638  E6D8                  OUT     0D8H
                                    ; STATEMENT # 654
063A  5D                    POP     BP
063B  C20400                RET     4H
                OVC         ENDP
                                    ; STATEMENT # 655
                UCCR        PROC NEAR
063E  55                    PUSH    BP
063F  8BEC                  MOV     BP,SP
0641  51                    PUSH    CX
                                    ; STATEMENT # 659
0642  E80000                CALL    FLAGS
0645  8946FE                MOV     [BP].TFLAGS,AX
                                    ; STATEMENT # 660
0648  FA                    CLI
                                    ; STATEMENT # 661
0649  8A4606                MOV     AL,[BP].REG
064C  E6DC                  OUT     0DCH
                                    ; STATEMENT # 662
```

```
DEFN  ADDR   SIZE  NAME, ATTRIBUTES, AND REFERENCES

612   0004H     1  BACK . . . . . .   BYTE IN PROC (SCA) PARAMETER AUTOMATIC        612  615
 34                BEL. . . . . . .   LITERALLY '007H'       468
161   0000H     1  BIIL . . . . . .   BYTE EXTERNAL(36)
 67                BLACK. . . . . .   LITERALLY '0$0$0B'           342  645
 68                BLUE . . . . . .   LITERALLY '0$0$1B'
  4                BOOL . . . . . .   LITERALLY 'BYTE'       165  170  171  173  175  292  293  312
                                        443
141   0000H    83  BOOT_PAR . . . . .   STRUCTURE EXTERNAL(16)
      0000H     1      INDEX. . . . .   BYTE
      0001H     1      PORT . . . . .   BYTE
      0002H    80      STRNG. . . . .   BYTE ARRAY(80)
      0052H     1      UNIT . . . . .   BYTE
 35                BS . . . . . . .   LITERALLY '008H'       470
162   0000H     1  BSIL . . . . . .   BYTE EXTERNAL(37)
                  BUILDPTR . . . . .   BUILTIN          347  356  357
 54   0000H     4  BYTEPTR. . . . . .   POINTER IN PROC (INCB) PARAMETER          54
500   0004H     1  C. . . . . . . .   BYTE IN PROC (D_KBD) PARAMETER AUTOMATIC      500  502
508   0004H     1  C. . . . . . . .   BYTE IN PROC (TXMTC) PARAMETER AUTOMATIC      508  510
464   0004H     1  C. . . . . . . .   BYTE IN PROC (D_CRT) PARAMETER AUTOMATIC      464  466  468  470
                                        472  474  480  490
 49   0000H     1  C. . . . . . . .   BYTE IN PROC (DXMTC) PARAMETER      49
506   0019H     1  C. . . . . . . .   BYTE IN PROC (TXMT)          516*  518  542  543  544
 57   0000H     1  C. . . . . . . .   BYTE IN PROC (SXMTC) PARAMETER      57
506   001AH     1  CA . . . . . . .   BYTE IN PROC (TXMT)          517*  526  534  536  539  540
 36                CAN. . . . . . .   LITERALLY '018H'
130   0000H     1  CHAR . . . . . .   BYTE IN PROC (WRITEC) PARAMETER          130
125   0000H     1  CHAR . . . . . .   BYTE IN PROC (MCU) PARAMETER       125
177   0000H     1  CHAR . . . . . .   BYTE IN PROC (S_CRT) PARAMETER     177
298   0000H     1  CMND . . . . . .   BYTE IN PROC (WRITK) PARAMETER     298
135                CODE_SEG . . . . .   LITERALLY '0FE05H'
180   0000H     1  COL. . . . . . .   BYTE IN PROC (UCA) PARAMETER       180
639   0006H     1  COL. . . . . . .   BYTE IN PROC (XMTSC) PARAMETER AUTOMATIC      639  642
169   0000H     7  COLOR. . . . . .   STRUCTURE EXTERNAL(41)
      0000H     1      FORE . . . . .   BYTE       614*  616  618  619  620
      0001H     1      BACK . . . . .   BYTE       615*  616  618  619  621
      0002H     1      MASK . . . . .   BYTE       616*  617
      0003H     1      CLEAR. . . . .   BYTE       618*
      0004H     1      PAINTED. . . .   BYTE       619*  620  621
      0005H     1      FONT . . . . .   BYTE       620*
      0006H     1      COMP_FONT. . .   BYTE       621*
639   0004H     2  COUNT. . . . . .   WORD IN PROC (XMTSC) PARAMETER AUTOMATIC      639  646
 37                CR . . . . . . .   LITERALLY '00DH'       437  480
122   0000H        CRLF . . . . . .   PROCEDURE EXTERNAL(8) STACK=0000H       494
170   0000H     3  CRTC_CURSOR. . . .   STRUCTURE EXTERNAL(42)
      0000H     2      START. . . . .   WORD       418  420  599*
      0002H     1      UPDATE . . . .   BYTE       397*  415  421*  444  600*
171   0000H     3  CRTC_DISPLAY . . .   STRUCTURE EXTERNAL(43)
      0000H     2      START. . . . .   WORD       388*  410  412  573*  573  588*  588  599
      0002H     1      UPDATE . . . .   BYTE       397*  407  413*  444  574*  589*
328   0000H    16  CRTC_REG_50. . . .   BYTE ARRAY(16) DATA          334
329   0010H    16  CRTC_REG_60. . . .   BYTE ARRAY(16) DATA          335
 75                CRTC_RSEL. . . . .   LITERALLY 'IO_CRTC+0'        409  411  417  419  661
```

```
140  0000H     2  DS_SIZE. . . . . .  WORD EXTERNAL(15)
 43  0000H        DXMTC. . . . . .  PROCEDURE EXTERNAL(0) STACK=0000H  502
463  02A6H   198  D_CRT. . . . . .  PROCEDURE PUBLIC STACK=0010H
499  036CH    19  D_KBD. . . . . .  PROCEDURE PUBLIC STACK=0008H
385  0166H    96  D_TTY_INIT . . . .  PROCEDURE STACK=0006H  455
446  026FH    55  D_TTY_RES. . . . .  PROCEDURE PUBLIC STACK=000CH
144  0000H     4  D_XMTC . . . . . .  POINTER EXTERNAL(19)
145  0000H     4  EDC. . . . . .  POINTER EXTERNAL(20)  570  584
146  0000H     4  EMEC . . . . . .  POINTER EXTERNAL(21)
605  0561H    16  ENABLE_LINES . . .  PROCEDURE PUBLIC STACK=000AH
 39              EOL. . . . . .  LITERALLY '080H'
 40              EOS. . . . . .  LITERALLY '030H'
 41              ESC. . . . . .  LITERALLY '01BH'  521  529  537
172  0000H    10  ESCP . . . . . .  STRUCTURE EXTERNAL(44)
     0000H     1      COMMAND. . . .  BYTE
     0001H     2      FUNCTION . . .  WORD
     0003H     1      MODE . . . . .  BYTE
     0004H     1      OPER_COUNT . .  BYTE
     0005H     1      OPER_INDEX . .  BYTE
     0006H     4      OPERAND. . . .  BYTE ARRAY(4)
198                  F86_AF . . . . . .  LITERALLY '0000$0001$0000B'
200                  F86_CF . . . . . .  LITERALLY '0000$0000$0001B'
193                  F86_DF . . . . . .  LITERALLY '0100$0000$0000B'
194                  F86_IF . . . . . .  LITERALLY '0010$0000$0000B'  439  452  459  575  590  601
                                          663
192                  F86_OF . . . . . .  LITERALLY '1000$0000$0000B'
199                  F86_PF . . . . . .  LITERALLY '0000$0000$0100B'
196                  F86_SF . . . . . .  LITERALLY '0000$1000$0000B'
195                  F86_TF . . . . . .  LITERALLY '0001$0000$0000B'
197                  F86_ZF . . . . . .  LITERALLY '0000$0100$0000B'
  5                  FALSE. . . . . .  LITERALLY '000H'  341  390  395  397  413  421  644
 42                  FF . . . . . .  LITERALLY '00CH'
 51  0000H           FLAGS. . . . . .  PROCEDURE WORD EXTERNAL(1) STACK=0000H  405  449  456  571
                                          586  597  659
147  0000H     4  FONT . . . . . .  POINTER EXTERNAL(22)
156  0000H     2  FONTSIZE . . . . .  WORD EXTERNAL(31)
612  0006H     1  FORE . . . . . .  BYTE IN PROC (SCA) PARAMETER AUTOMATIC  612  614
303                  GEN_CNT. . . . . .  LITERALLY 'IO_GENERAL+1'
304                  GEN_DAT. . . . . .  LITERALLY 'IO_GENERAL'
305                  GEN_DIR. . . . . .  LITERALLY 'IO_GENERAL'
 71                  GREEN. . . . . .  LITERALLY '1$0$0B'
173  0000H    18  H19_MODE . . . . .  STRUCTURE EXTERNAL(45)
     0000H     1      ALTERNATE. . .  BYTE
     0001H     1      ANSI . . . . .  BYTE
     0002H     1      AUTO_CR. . . .  BYTE  390*  476
     0003H     1      AUTO_LF. . . .  BYTE  390*  483
     0004H     1      AUTO_REPEAT. .  BYTE
     0005H     1      BWO. . . . . .  BYTE  617*  617
     0006H     1      CURSOR . . . .  BYTE  389*  627  628  634*  634
     0007H     1      CURSOR_ON. . .  BYTE  391*  626
     0008H     1      EXPAND . . . .  BYTE
     0009H     1      GRAPHIC. . . .  BYTE
     000AH     1      INSERT . . . .  BYTE  390*  488
     000BH     1      KEY_EN . . . .  BYTE  391*
     000CH     2      REVERSE. . . .  WORD  388*  490
     000EH     1      SHIFTED. . . .  BYTE
```

```
  7                 IO_DIP . . . . . LITERALLY '00FFH'       333
 24                 IO_GENERAL . . . . LITERALLY '00E0H'
  7                 IO_HIADR . . . . . LITERALLY '00FDH'
 18                 IO_INT_MS. . . . . LITERALLY '00F2H'       451  458
 19                 IO_INT_SL. . . . . LITERALLY '00F0H'
 17                 IO_KEYBRD. . . . . LITERALLY '00F4H'
 26                 IO_LTPN. . . . . LITERALLY '00DEH'
 10                 IO_MEM . . . . . LITERALLY '00FCH'
 23                 IO_PRINTER . . . . LITERALLY '00E2H'
 12                 IO_RSRV_1. . . . . LITERALLY '00FAH'
 14                 IO_RSRV_2. . . . . LITERALLY '00F8H'
 15                 IO_RSRV_3. . . . . LITERALLY '00F7H'
 16                 IO_RSRV_4. . . . . LITERALLY '00F6H'
 25                 IO_RSRV_5. . . . . LITERALLY '00DFH'
 29                 IO_RSRV_6. . . . . LITERALLY '00C0H'
 21                 IO_SER_A . . . . . LITERALLY '00E8H'
 20                 IO_SER_B . . . . . LITERALLY '00ECH'
 13                 IO_SOUND . . . . . LITERALLY '00F9H'
  8                 IO_SWAP. . . . . LITERALLY '00FEH'
 22                 IO_TIMER . . . . . LITERALLY '00E4H'
 11                 IO_TSTAT . . . . . LITERALLY '00FBH'
 28                 IO_VIDEO . . . . . LITERALLY '00D8H'       340  374  375  376  377  378  379  380
                                                                381  382  569  585  652
 31                 IO_Z207_0. . . . . LITERALLY '00B0H'
 30                 IO_Z207_1. . . . . LITERALLY '00B8H'
 33                 IO_Z217_0. . . . . LITERALLY '00A8H'
 32                 IO_Z217_1. . . . . LITERALLY '00ACH'
 65  0000H      1   IVI. . . . . BYTE IN PROC (INIT_ITV) PARAMETER          65
 65  0000H      4   IVP. . . . . POINTER IN PROC (INIT_ITV) PARAMETER       65
261                 KC_ARF . . . . . LITERALLY '02H'
262                 KC_ARO . . . . . LITERALLY '01H'
263                 KC_BEP . . . . . LITERALLY '07H'         469
268                 KC_CFI . . . . . LITERALLY '05H'
269                 KC_CLK . . . . . LITERALLY '06H'
264                 KC_DI. . . . . LITERALLY '0DH'
265                 KC_EI. . . . . LITERALLY '0CH'         396
270                 KC_KBD . . . . . LITERALLY '09H'
271                 KC_KBE . . . . . LITERALLY '08H'
266                 KC_KCF . . . . . LITERALLY '04H'
267                 KC_KCO . . . . . LITERALLY '03H'
272                 KC_MNS . . . . . LITERALLY '0BH'
273                 KC_MUD . . . . . LITERALLY '0AH'
274                 KC_RES . . . . . LITERALLY '00H'         392
292  0000H      2   KEY. . . . . STRUCTURE EXTERNAL(55)
     0000H      1       CHAR . . . . . BYTE
     0001H      1       AVAIL. . . . . BYTE       395*
256                 KEY_CMND . . . . . LITERALLY 'IO_KEYBRD+1'
257                 KEY_DATA . . . . . LITERALLY 'IO_KEYBRD+0'
258                 KEY_STAT . . . . . LITERALLY 'IO_KEYBRD+1'
157  0000H    256   KMAP . . . . . BYTE ARRAY(256) EXTERNAL(32)       502
259                 KS_CXRDY . . . . . LITERALLY '001B'
260                 KS_RXRDY . . . . . LITERALLY '010B'
286                 KY_BREAK . . . . . LITERALLY '0AAH'
282                 KY_DOWN. . . . . LITERALLY '0A6H'
275                 KY_ENTER . . . . . LITERALLY '08DH'
277                 KY_F0. . . . . LITERALLY '096H'
```

```
238                OCW1_M4. . . . . LITERALLY '0001$0000B'
237                OCW1_M5. . . . . LITERALLY '0010$0000B'
236                OCW1_M6. . . . . LITERALLY '0100$0000B'        451  458
235                OCW1_M7. . . . . LITERALLY '1000$0000B'
246                OCW2_EOI . . . . LITERALLY '0010$0000B'
250                OCW2_L0. . . . . LITERALLY '0000$0001B'
249                OCW2_L1. . . . . LITERALLY '0000$0010B'
248                OCW2_L2. . . . . LITERALLY '0000$0100B'
247                OCW2_OCW2. . . . LITERALLY '0000$0000B'
244                OCW2_ROT . . . . LITERALLY '1000$0000B'
245                OCW2_SEOI. . . . LITERALLY '0100$0000B'
251                OCW3_ESMM. . . . LITERALLY '0100$0000B'
253                OCW3_OCW3. . . . LITERALLY '0000$1000B'
254                OCW3_POLL. . . . LITERALLY '0000$0100B'
252                OCW3_SMM . . . . LITERALLY '0010$0000B'
255                OCW3_SRIS. . . . LITERALLY '0000$0010B'
                   OUTPUT . . . . . BUILTIN          340*  374*  375*  376*  377*  378*  379*  380*  381*
                                     382*  409*  410*  411*  412*  417*  418*  419*  420*  451*  458*  569*
                                     585*  652*  661*  662*
649  0622H    28   OVC. . . . . . . PROCEDURE PUBLIC STACK=0006H      346  368  369
332  0000H     1   P1 . . . . . . . BYTE BASED(P1PTR) IN PROC (INIT_VIDEO)        348  349  350  354*
                                     358  360  362  365*
332  0000H     4   P1PTR. . . . . . POINTER IN PROC (INIT_VIDEO)      334*  335*  337  347*  348  349
                                     350  356*  358  360  362
332  0000H     1   P2 . . . . . . . BYTE BASED(P2PTR) IN PROC (INIT_VIDEO)        359  360*  361  362
                                     366*
332  0004H     4   P2PTR. . . . . . POINTER IN PROC (INIT_VIDEO)      357*  359  361  362
332  0000H     1   P3 . . . . . . . BYTE BASED(P1PTR) ARRAY(1) IN PROC (INIT_VIDEO)     337
318                P6821_A. . . . . LITERALLY '0'
319                P6821_B. . . . . LITERALLY '2'
327                P6821_C1_0 . . . LITERALLY '001H'
326                P6821_C1_1 . . . LITERALLY '002H'
324                P6821_C2_0 . . . LITERALLY '008H'        374  376  378  381
323                P6821_C2_1 . . . LITERALLY '010H'        374  376  378  381
322                P6821_C2_2 . . . LITERALLY '020H'        376  378  381
320                P6821_IR01 . . . LITERALLY '030H'
321                P6821_IR02 . . . LITERALLY '040H'
325                P6821_ORA. . . . LITERALLY '004H'        374  378  381
300                PRN_CNT. . . . . LITERALLY 'IO_PRINTER+1'
301                PRN_DAT. . . . . LITERALLY 'IO_PRINTER'
302                PRN_DIR. . . . . LITERALLY 'IO_PRINTER'
150  0000H     4   PROMPT . . . . . POINTER EXTERNAL(25)
163  0000H     1   PSP. . . . . . . BYTE EXTERNAL(38)
134  0000H         P_ELIN . . . . . PROCEDURE EXTERNAL(51) STACK=0000H
18A  0000H         P_EPAG . . . . . PROCEDURE EXTERNAL(52) STACK=0000H      345
132  0000H         P_HOM. . . . . . PROCEDURE EXTERNAL(50) STACK=0000H      344  398
347  0460H    41   P_HT . . . . . . PROCEDURE STACK=0002H       473
357  0489H    36   P_LF . . . . . . PROCEDURE STACK=0004H       478  484
188  0000H         P_RES. . . . . . PROCEDURE EXTERNAL(53) STACK=0000H
151  0000H     4   RDC. . . . . . . POINTER EXTERNAL(26)        515
127  0000H         READC. . . . . . PROCEDURE BYTE EXTERNAL(10) STACK=0000H
395  0000H         READK. . . . . . PROCEDURE BYTE EXTERNAL(57) STACK=0000H      394
 59                RED. . . . . . . LITERALLY '0$1$0B'
656  0006H     1   REG. . . . . . . BYTE IN PROC (UCCR) PARAMETER AUTOMATIC      656  661
  62 0000H     4   REGP . . . . . . POINTER IN PROC (XEC) PARAMETER          62
164  0000H     4   REGP . . . . . . POINTER EXTERNAL(39)
```

```
      0020H          VIDEO. . . . . .   PROCEDURE STACK=0000H
   59 0000H          VIDEO_INTR_A . . .  PROCEDURE EXTERNAL(4) STACK=0000H
   79                VID_CDC. . . . . .  LITERALLY 'IO_VIDEO+1'        374  376  378
   78                VID_CDD. . . . . .  LITERALLY 'IO_VIDEO+0'        377
   77                VID_CMD. . . . . .  LITERALLY 'IO_VIDEO+0'        375  652
   80                VRMM_DAT . . . . .  LITERALLY 'IO_VIDEO+2'        340  382  569  585
   82                VRMM_MPC . . . . .  LITERALLY 'IO_VIDEO+3'        379  381
   81                VRMM_MPD . . . . .  LITERALLY 'IO_VIDEO+2'        380
   45                VT . . . . . .      LITERALLY '00BH'
   74                WHITE. . . . . .    LITERALLY '1$1$1B'            342  645
  138 0000H      5   WIP. . . . . .     BYTE ARRAY(5) EXTERNAL(13)
  129 0000H          WRITEC . . . . . .  PROCEDURE EXTERNAL(11) STACK=0000H
  132 0000H          WRITES . . . . . .  PROCEDURE EXTERNAL(12) STACK=0000H
  297 0000H          WRITK. . . . . .    PROCEDURE EXTERNAL(58) STACK=0000H      392  396  469
  506 000CH      2   XC . . . . . .      WORD IN PROC (TXMT)           515  516  517
  154 0000H      4   XCA. . . . . .      POINTER EXTERNAL(29)          427
  168                XCOLOR_BACK. . . .  LITERALLY '00$111$000B'       534  536  540
  167                XCOLOR_FORE. . . .  LITERALLY '00$000$111B'       534  536  539
   61 0000H          XEC. . . . . .      PROCEDURE EXTERNAL(5) STACK=0000H
  175 0000H     10   XMT. . . . . .      STRUCTURE EXTERNAL(47)
      0000H      1       BURST. . . .    BYTE        425
      0001H      2       BCOUNT . . .    INTEGER           425*  425  426  511*  511  643*
      0003H      2       COUNT. . . .    WORD        388*  423  426  428*  428  436  646*
      0005H      1       COL. . . . .    BYTE        429*  429  430  432*  515  642*
      0006H      1       ROW. . . . .    BYTE        433*  433  515  641*
      0007H      1       COLOR. . . .    BYTE        534  536*  645*
      0008H      1       GRAPHIC. . .    BYTE        518  520*  520  522  542  644*
      0009H      1       REVERSE. . .    BYTE        526  528*  528  530  644*
  638 05F8H     42   XMTSC. . . . . .    PROCEDURE PUBLIC STACK=0008H
   46                XOFF . . . . . .    LITERALLY '013H'
   47                XON. . . . . .      LITERALLY '011H'
  166                XREVERSE . . . . .  LITERALLY '10$000$000B'       526
   73                YELLOW . . . . . .  LITERALLY '1$1$0B'
```

MODULE INFORMATION:

```
     CODE AREA SIZE     = 0661H   1633D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 001CH     28D
     MAXIMUM STACK SIZE = 0010H     16D
     1758 LINES READ
     0 PROGRAM WARNINGS
     0 PROGRAM ERRORS
```

END OF PL/M-86 COMPILATION

```
        =   $SAVE
        =   $NOLIST
            $       INCLUDE (':F2:KEYDEF.H')
        =   $SAVE
        =   $NOLIST
            $       INCLUDE (':F2:KEYLIB.H')
        =   $SAVE
        =   $NOLIST

            $       INCLUDE (':F2:VIDEO.H')
        =   $SAVE
        =   $NOLIST

            $       SET(COMPRESS)

            /*
             *      Internal Terminal Emulation Modes
             */
186   1     DC      M_NRM           LT      '0'     /* Normal character display     */;
187   1     DC      M_ESC           LT      '1'     /* Looking for Escape command   */;
188   1     DC      M_OPR           LT      '2'     /* Accepting Sequence Operands  */;
189   1     DC      M_ANS           LT      '3'     /* ANSI-Mode Sequence Processing*/;

190   1     DECLARE         LINE_INDEX      BYTE    /* Scratch Line Index           */
                             PUBLIC;
191   1     DECLARE         SAVED_HORZ_CHAR BYTE    /* Saved Horizontal Char. pos.  */
                             PUBLIC;
192   1     DECLARE         SAVED_VERT_LINE BYTE    /* Saved Vertical Line pos.     */
                             PUBLIC;

193   1     DECLARE         ESC_TABLE_1(*)  WORD DATA(
                              .P_EAM,                /* Enter ANSI Mode              */
                              .P_EAKM,               /* Enter Alternate Key-Pad Mode */
                              .P_XAKM,               /* Exit  Alternate Key-Pad Mode */
                              .P_UIM,                /* Un-Implemented               */
                              .P_EICM,               /* Enter Insert Character Mode  */
                              .P_CUP,                /* Cursor Up                    */
                              .P_CDN,                /* Cursor Down                  */
                              .P_CRT,                /* Cursor Right                 */
                              .P_CLF,                /* Cursor Left                  */
                              .P_EPAG,               /* Erase Page                   */
                              .P_EGM,                /* Enter Graphics Mode          */
                              .P_XGM,                /* Exit Graphics Mode           */
                              .P_HOM,                /* Home the Cursor              */
                              .P_RLF,                /* Reverse Line Feed            */
                              .P_EEOP,               /* Erase to End-Of-Page         */
                              .P_EEOL,               /* Erase to End-Of-Line         */
                              .P_IL,                 /* Insert Line                  */
                              .P_DL,                 /* Delete Line                  */
                              .P_DC,                 /* Delete Character             */
                              .P_XICM);              /* Exit Insert Character Mode   */
194   1     DECLARE         ESC_TABLE_2(*)  WORD DATA (
                              .P_DCA,                /* Direct Cursor Addressing     */
```

```
196   1     S_TTY_INIT:     PROCEDURE               PUBLIC;
197   2         DO;
198   3         ESCP.MODE = M_NRM;
199   3         H19_MODE.EXPAND = TRUE;
200   3         H19_MODE.ANSI,
                H19_MODE.GRAPHIC,
                H19_MODE.ALTERNATE,
                H19_MODE.SHIFTED = FALSE;
201   3         CALL P_SCP;                   /* Initialize Saved Cursor Position     */
202   3         END;
203   2       END;



204   1     S_CRT:          PROCEDURE(char)         PUBLIC;
205   2       DECLARE       char            BYTE;
206   2         DO;
207   3         IF  XMT.COUNT = 0  THEN
208   3           DO;
209   4           char = char AND 07FH;
210   4           DO CASE  ESCP.MODE  ;

                  /*
                   *      Normal Character
                   */
211   5             DO;
212   6             IF       char <> ESC    THEN
213   6                      DO;
214   7                      IF  H19_MODE.GRAPHIC AND ('^'<=char AND char<DEL)
                               THEN
215   7                        CALL D_CRT(char-'^'+DEL+1);
216   7                      ELSE
217   7                        CALL D_CRT(char);
                             END;
218   6             ELSE IF  H19_MODE.ANSI  THEN
219   6                      ESCP.MODE = M_ANS;
220   6             ELSE
                             ESCP.MODE = M_ESC;
221   6             END;

                  /*
                   *      Escape Sequence Command
                   */
222   5             DO;
223   6             ESCP.COMMAND = char ;
224   6             ESCP.OPER_COUNT = DOC(char) ;
225   6             ESCP.OPER_INDEX = 0 ;
226   6             ESCP.MODE = M_OPR ;
227   6             IF  ESCP.OPER_COUNT = 0  THEN
228   6               CALL PES;
229   6             END;

                  /*
                   *      Escape Sequence Operands
```

```
                /*
                 *      DCA     - Direct Cursor Addressing
                 *
                 *          'DCA' performs the direct cursor addressing
                 *          function.  This involves setting the cursor
                 *          to the specified address.  This cursor will
                 *          NOT be moved to the status line if the status
                 *          line is not enabled.  Any illegal line spec-
                 *          ification results in the cursor remaining at
                 *          the current line.  Any illegal column spec-
                 *          ification results in positioning the cursor
                 *          at the end of the current line.
                 *
                 *      Parameters:
                 *
                 *          row - Vertical Row [0,h19_par.sli]
                 *          col - Horizontal Column [0,h19_par.cpl-1]
                 *
                 */

244   1         DCA:              PROCEDURE(row,col)      PUBLIC;
245   2           DECLARE         row                 BYTE,
                                  col                 BYTE;
246   2             DO;
247   3             IF  (0 <= row  AND  row < H19_PAR.SLI) OR
                      (row=H19_PAR.SLI AND H19_MODE.STATUS)  THEN
248   3               VERT_LINE = row;
249   3             IF  (0 <= col  AND  col < H19_PAR.CPL)
                      THEN
250   3                 HORZ_CHAR = col;
251   3                ELSE
                        HORZ_CHAR = H19_PAR.CPL-1;
252   3             END;
253   2           END;



                /*
                 *      DOC     - Determine Operand Count
                 *
                 *          DOC determines the operand count for the specified escape
                 *          sequence.
                 *
                 *      Parameters:
                 *
                 *          cmd         = Escape Sequence or Command to return
                 *                        the number of parameters for.
                 *
                 *      Exit:
                 *
                 *          Returns the number of parameters (characters) to be
```

```
                /*
                 *      PES     - Process Escape Sequence
                 *
                 *      Description
                 *
                 *          PES processes the current escape sequence.  When PES
                 *          is invoked, it is assumed that all of the required
                 *          parameters/operands have been input.
                 *
                 *      Entry:
                 *
                 *          ESCP.COMMAND        = Escape Sequence Command
                 *          ESCP.OPERAND(i)     = Operands for this escape sequence
                 *
                 *      Exit:
                 *
                 *          Emulation mode, 'ESCP.MODE',  reset to normal character mode
                 *
                 */

266   1         PES:            PROCEDURE;
267   2           DO;
268   3             ESCP.MODE = M_NRM;
269   3             IF       '<'<=ESCP.COMMAND  AND  ESCP.COMMAND<='O'  THEN
270   3                      ESCP.FUNCTION = ESC_TABLE_1(ESCP.COMMAND-'<');
271   3             ELSE IF  'Y'<=ESCP.COMMAND  AND  ESCP.COMMAND<='_'  THEN
272   3                      ESCP.FUNCTION = ESC_TABLE_2(ESCP.COMMAND-'Y');
273   3             ELSE IF  ESCP.COMMAND = 'b'                         THEN
274   3                      ESCP.FUNCTION = .P_EBOP;
275   3             ELSE IF  'i'<=ESCP.COMMAND  AND  ESCP.COMMAND<=')'  THEN
276   3                      ESCP.FUNCTION = ESC_TABLE_3(ESCP.COMMAND-'i');
277   3             ELSE IF  '#' = ESCP.COMMAND                         THEN
278   3                      ESCP.FUNCTION = .P_XMTP;
279   3             ELSE
                               ESCP.FUNCTION = .P_UIM;
280   3             CALL ESCP.FUNCTION;
281   3             CALL CURSOR;
282   3             END;
283   2           END;



                $EJECT
```

```
                /*
                 *      P_UIM   - Un-Implemented Escape Sequence
                 *
                 *          'P_UIM' processes the Un-Implemented Escape
                 *          sequences.  Eventually these will be vectored
                 *          through RAM.
                 *
                 */

290   1         P_UIM:          PROCEDURE               PUBLIC;
291   2             DO;
292   3             CALL UIES;
293   3             END;
294   2           END;


                $EJECT
```

```
346   2         END;

                $IF EXTENDED_MONITOR
                P_EHSM:           PROCEDURE EXTERNAL;
                  END;
                P_XHSM:           PROCEDURE EXTERNAL;
                  END;
                $ENDIF

347   1         P_DK:             PROCEDURE EXTERNAL;
348   2           END;
349   1         P_EK:             PROCEDURE EXTERNAL;
350   2           END;
351   1         P_EKPS:           PROCEDURE EXTERNAL;
352   2           END;
353   1         P_XKPS:           PROCEDURE EXTERNAL;
354   2           END;
355   1         P_ERVM:           PROCEDURE EXTERNAL;
356   2           END;
357   1         P_XRVM:           PROCEDURE EXTERNAL;
358   2           END;
359   1         P_EWRAP:          PROCEDURE EXTERNAL;
360   2           END;
361   1         P_XWRAP:          PROCEDURE EXTERNAL;
362   2           END;

                $IF EXTENDED_MONITOR
                P_BAUD:           PROCEDURE EXTERNAL;
                  END;
                $ENDIF

363   1         P_RES:            PROCEDURE EXTERNAL;
364   2           END;
365   1         P_RM:             PROCEDURE EXTERNAL;
366   2           END;
367   1         P_SM:             PROCEDURE EXTERNAL;
368   2           END;
369   1         P_COLOR:          PROCEDURE EXTERNAL;
370   2           END;
371   1         P_IDENT:          PROCEDURE EXTERNAL;
372   2           END;
373   1         P_IVT52:          PROCEDURE EXTERNAL;
374   2           END;
375   1         P_XMTCC:          PROCEDURE EXTERNAL;
376   2           END;
377   1         P_XMTCL:          PROCEDURE EXTERNAL;
378   2           END;
379   1         P_XMTSL:          PROCEDURE EXTERNAL;
380   2           END;
381   1         P_XMTP:           PROCEDURE EXTERNAL;
382   2           END;

                $       ELSE

                $       EJECT
```

```
/*
 *      P_CUP   - Perform Cursor Up
 *
 *          P_CUP moves the cursor up one line.  This will NOT
 *          move the cursor off of the status line.  If the
 *          cursor is currently on line-0, then no action is
 *          taken.
 *
 */

P_CUP:          PROCEDURE;
    DO;
    IF  VERT_LINE < H19_PAR.SLI  THEN
      CALL DCA(VERT_LINE-1,HORZ_CHAR);
    END;
  END;



/*
 *      P_CPR   - Cursor Position Report
 *
 *          'P_CPR' reports the cursor position in the format
 *          expected by the Direct Cursor Addressing escape
 *          sequence.
 *
 */

P_CPR:          PROCEDURE;
    DO;
    CALL TEC;
    CALL S$XMTC('Y');
    CALL S$XMTC(VERT_LINE+' ');
    CALL S$XMTC(HORZ_CHAR+' ');
    END;
  END;



/*
 *      P_DCA   - Perform Director Cursor Addressing
 *
 *          P_DCA sets the cursor to the specified address.
 *          this routine invokes the 'DCA' routine to per-
 *          form the cursor addressing, and assumes that
 *          the operands to the escape sequence are in the
 *          operand buffer.  The first byte is assumed to
 *          be the row, and the second the column.  The bytes
 *          are as in the H-19/VT-52, which means that the
 *          values are offset by the value of the space char-
```

```
P_RLF:          PROCEDURE;
    DO;
    IF          VERT_LINE = H19_PAR.SLI  THEN
                RETURN;
    ELSE IF     VERT_LINE <> 0           THEN
                VERT_LINE = VERT_LINE - 1 ;
    ELSE
                CALL REV_SCROLL;
    END;
  END;



/*
 *      P_SCP   - Save Cursor Position
 *
 *          P_SCP saves the current cursor position so that
 *          it may be restored with the appropriate escape
 *          sequence.  It is important to note that the be-
 *          haviour to be emulated stipulates that the pos-
 *          itions NOT be stacked, therefore, only one set
 *          of save locations is used.
 *
 */

P_SCP:          PROCEDURE;
    DO;
    SAVED_HORZ_CHAR = HORZ_CHAR ;
    SAVED_VERT_LINE = VERT_LINE ;
    END;
  END;



$EJECT



/*
 *      P_EBOL  - Process Erase to Beginning Of Line
 *
 *          P_EBOL erases to the beginning of the current line,
 *          including the current character.
 *
 */

P_EBOL:         PROCEDURE;
    DO;
    CALL EDC(VERT_LINE,0,HORZ_CHAR+1);
    END;
  END;
```

```
      END;
    END;
  END;



/*
 *      P_ELIN  - Process Erase Line
 *
 *          P_ELIN erases the entire current line.
 *
 */

P_ELIN:         PROCEDURE               PUBLIC;
    DO;
    CALL EDL(VERT_LINE);
    END;
  END;



/*
 *      P_EPAG  - Erase Page
 *
 *          P_EPAG erases the entire page.
 *
 */

P_EPAG:         PROCEDURE               PUBLIC;
    DO;
    IF  VERT_LINE = H19_PAR.SLI
      THEN
        CALL P_ELIN;
      ELSE
        DO;
        DO  LINE_INDEX=0 TO H19_PAR.SLI-1 ;
          CALL EDL(LINE_INDEX);
          END;
        CALL P_HOM;
        IF  NOT H19_MODE.STATUS  THEN
          H19_MODE.BWO = (COLOR.MASK = 7 );
        END;
    END;
  END;



/*
 *      P_DC    - Perform Delete Character
 *
 *          P_DC deletes the character at the current character
 *          position.  This is done by moving the rest of the
 *          line to the current character position, and deleting
```

```
        THEN
          DO;
          DO  LINE_INDEX = VERT_LINE+1 TO H19_PAR.SLI-1  ;
            CALL MDL(LINE_INDEX,LINE_INDEX-1);
            END;
          CALL EDL(H19_PAR.SLI-1);
          END;
        ELSE
          CALL P_ELIN;
      CALL DCA(VERT_LINE,0);
      END;
    END;
$       ENDIF



/*
 *      P_IL    - Perform Insert Line
 *
 *          P_IL performs the insert line function.  A blank line
 *          line is inserted at the current cursor position after
 *          the remaining lines have been moved down.
 *
 */

$       IF 0=1
P_IL:           PROCEDURE;
    DO;
    IF  VERT_LINE=H19_PAR.SLI  THEN
      DO;
      CALL P_ELIN;
      RETURN;
      END;

    IF  VERT_LINE < H19_PAR.SLI/2
      THEN
        DO;
        CALL REV_SCROLL;
        DO  LINE_INDEX=1 TO VERT_LINE ;
          CALL MDL(LINE_INDEX,LINE_INDEX-1);
          END;
        END;
      ELSE
        DO;
        LINE_INDEX=H19_PAR.SLI-1;
        DO WHILE  LINE_INDEX <> VERT_LINE  ;
          CALL MDL(LINE_INDEX-1,LINE_INDEX);
          LINE_INDEX=LINE_INDEX-1;
          END;
        END;
    CALL P_ELIN;
    END;
  END;
$       ELSE
P_IL:           PROCEDURE;
```

```
/*
 *      P_EAM   - Enter ANSI Mode
 *
 *          'P_EAM' enters ANSI Mode, which means that the
 *          extended mode flag is set for external user
 *          processing.
 *
 */

P_EAM:          PROCEDURE;
    DO;
    H19_MODE.ANSI = TRUE;
    END;
  END;



/*
 *      P_EICM  - Process Enter Insert Character Mode
 *
 *          P_EICM processes the insert character mode escape
 *          sequence.  It merely sets the global boolean to
 *          true.
 *
 */

P_EICM:         PROCEDURE;
    DO;
    H19_MODE.INSERT = TRUE;
    END;
  END;



/*
 *      P_XICM  - Process Exit Insert Character Mode
 *
 *          P_XICM processes the Exit insert character mode
 *          escape sequence.  It merely sets the global bool-
 *          ean to FALSE.
 *
 */

P_XICM:         PROCEDURE;
    DO;
    H19_MODE.INSERT = FALSE;
    END;
  END;
```

```
 *       P_XHSM  - Exit Hold-Screen Mode
 *
 *            'P_XHSM' exits the Hold-Screen Mode by setting the
 *            boolean H19_MODE.HOLD to FALSE.
 *
 */

$        IF EXTENDED_MONITOR
P_XHSM:           PROCEDURE;
    DO;
    END;
  END;
$        ENDIF



/*
 *       P_DK    - Disable Keyboard
 *
 *            'P_DK' disables the key-board.
 *
 */

P_DK:             PROCEDURE;
    DO;
    CALL WRITK(KC_KBD);
    H19_MODE.KEY_EN = FALSE;
    END;
  END;



/*
 *       P_EK    - Enable Key-Board
 *
 *            'P_EK' enables the key-board.
 *
 */

P_EK:             PROCEDURE;
    DO;
    CALL WRITK(KC_KBE);
    H19_MODE.KEY_EN = TRUE;
    END;
  END;



/*
 *       P_EKPS  - Enter Key-Pad Shifted Mode
 *
 *            'P_EKPS' enters the key-pad shifted mode by
 *            merely setting the boolean 'H19_MODE.SHIFTED'
```

```
 *          zeroes for invoking the display font character
 *          procedure.
 *
 */

P_XRVM:          PROCEDURE;
    DO;
    H19_MODE.REVERSE = 00000H;
    END;
  END;



/*
 *      P_EWRAP - Enter Wrap Mode
 *
 *          'P_EWRAP' enters the H19 character wrap mode
 *          by setting the global H19_MODE.WRAP' to TRUE.
 *
 */

P_EWRAP:         PROCEDURE;
    DO;
    H19_MODE.WRAP = TRUE;
    END;
  END;



/*
 *      P_XWRAP - Exit Wrap Mode
 *
 *          'P_XWRAP' exits the H19 character wrap mode
 *          by setting the global H19_MODE.WRAP to FALSE.
 *
 */

P_XWRAP:         PROCEDURE;
    DO;
    H19_MODE.WRAP = FALSE;
    END;
  END;



$EJECT



/*
 *      P_BAUD  - Process Baud Rate
 *
```

```
*               >       -
*               ?       - Disable Function-Key Expansion
*               @       - Disable Key-Board Up/Down Mode
*
*/

P_RM:           PROCEDURE;
    DO;
    IF  '1'<=ESCP.OPERAND(0) AND ESCP.OPERAND(0)<='@'  THEN
      DO CASE  ESCP.OPERAND(0)-'1'  ;

        DO;                                 /* 1 */
        H19_MODE.STATUS = FALSE;
        CALL EDL(H19_PAR.SLI);
        CALL ENABLE_LINES(H19_PAR.SLI);
        END;

        CALL WRITK(KC_KCO);                 /* 2 */

        CALL P_UIM;                         /* 3 */

        CALL SCP1(NOT CSR_CSD,              /* 4 */
                  H19_PAR.SPC-3);

        DO;                                 /* 5 */
        H19_MODE.CURSOR_ON = TRUE;
        CALL SCP;
        END;

        CALL P_XKPS;                        /* 6 */

        CALL P_XAKM;                        /* 7 */

        H19_MODE.AUTO_LF = FALSE;           /* 8 */

        H19_MODE.AUTO_CR = FALSE;           /* 9 */

        CALL P_UIM;                         /* : */

        CALL SCP1(CSR_CSD,                  /* ; */
                  CSR_BLINK16);

        CALL WRITK(KC_ARO);                 /* < */

        CALL P_UIM;                         /* = */

        CALL P_UIM;                         /* > */

        H19_MODE.EXPAND = FALSE;            /* ? */

        DO;                                 /* @ */
        H19_MODE.UPDN = FALSE;
        CALL WRITK(KC_MNS);
        END;

        END;
```

```
            CALL P_EKPS;                        /* 6 */

            CALL P_EAKM;                        /* 7 */

            H19_MODE.AUTO_LF = TRUE;            /* 8 */

            H19_MODE.AUTO_CR = TRUE;            /* 9 */

            CALL P_UIM;                         /* : */

            CALL SCP1(NOT CSR_BLINK,0);         /* ; */

            CALL WR1TK(KC_ARF);                 /* < */

            CALL P_UIM;                         /* = */

            CALL P_UIM;                         /* > */

            H19_MODE.EXPAND = TRUE;             /* ? */

            DO;                                 /* @ */
            H19_MODE.UPDN = TRUE;
            CALL WR1TK(KC_MUD);
            END;

            END;
        END;
      END;


$EJECT


P_COLOR:        PROCEDURE;
    DO;
    IF  ('0'<=ESCP.OPERAND(0) AND ESCP.OPERAND(0)<='7') AND
        ('0'<=ESCP.OPERAND(1) AND ESCP.OPERAND(1)<='7')       THEN
      CALL SCA(ESCP.OPERAND(0)-'0',ESCP.OPERAND(1)-'0');
    END;
  END;


P_IDENT:        PROCEDURE;
    DO;
    IF  ESCP.OPERAND(0) = '0'  THEN
      DO;
      CALL TEC;
      CALL S$XMTC('i');
      CALL S$XMTC('E');
```

```
                    CALL XMTSC(H19_PAR.SLI,0,H19_PAR.CPL);
                  ELSE
                    CALL S$XMTC(CR);
                END;
              END;



            P_XMTP:         PROCEDURE;
                DO;
                CALL XMTSC(0,0,H19_PAR.CPL*(H19_PAR.LPS-1));
                END;
              END;



            $       ENDIF


383   1     END;
```

```
00BD  B15E            MOV     CL,5EH
00BF  8A4604          MOV     AL,[BP].CHAR
00C2  3AC8            CMP     CL,AL
00C4  770F            JA      @6
00C6  B27F            MOV     DL,7FH
00C8  3AC2            CMP     AL,DL
00CA  7309            JNB     @6
                                      ; STATEMENT # 215
00CC  2AC1            SUB     AL,CL
00CE  02C2            ADD     AL,DL
00D0  FEC0            INC     AL
00D2  50              PUSH    AX      ; 1
00D3  EB03            JMP     @2
                                      ; STATEMENT # 216
                @6:
00D5  FF7604          PUSH    [BP].CHAR; 1
                @2:
00D8  E80000          CALL    D_CRT
                                      ; STATEMENT # 218
                @8:
00DB  EB57            JMP     @1
                @5:
00DD  A00100          MOV     AL,H19_MODE+1H
00E0  D0D8            RCR     AL,1
00E2  7307            JNB     @9
                                      ; STATEMENT # 219
00E4  C606030003      MOV     ESCP+3H,3H
                                      ; STATEMENT # 220
00E9  EB49            JMP     @1
                @9:
00EB  C606030001      MOV     ESCP+3H,1H
                                      ; STATEMENT # 222
00F0  EB42            JMP     @1
                @11:
                                      ; STATEMENT # 223
00F2  8A4604          MOV     AL,[BP].CHAR
00F5  A20000          MOV     ESCP,AL
                                      ; STATEMENT # 224
00F8  50              PUSH    AX      ; 1
00F9  E87F00          CALL    DOC
00FC  A20400          MOV     ESCP+4H,AL
                                      ; STATEMENT # 225
00FF  C606050000      MOV     ESCP+5H,0H
                                      ; STATEMENT # 226
0104  C606030002      MOV     ESCP+3H,2H
                                      ; STATEMENT # 227
0109  08C0            OR      AL,AL
010B  EB17            JMP     @7
                @13:
                                      ; STATEMENT # 231
010D  8A1E0500        MOV     BL,ESCP+5H
0111  B700            MOV     BH,0H
0113  8A4604          MOV     AL,[BP].CHAR
0116  88870600        MOV     ESCP[BX+6H],AL
                                      ; STATEMENT # 232
011A  FEC3            INC     BL
```

```
0239  55              PUSH    BP
023A  8BEC            MOV     BP,SP
                                        ; STATEMENT # 287
023C  FF7604          PUSH    [BP].LINE; 1
023F  B000            MOV     AL,0H
0241  50              PUSH    AX        ; 2
0242  FF360000        PUSH    H19_PAR ; 3
0246  FF1E0000        CALL    EDC
                                        ; STATEMENT # 289
024A  5D              POP     BP
024B  C20200          RET     2H
            EDL       ENDP
                                        ; STATEMENT # 290
            P_UIM       PROC NEAR
024E  55              PUSH    BP
024F  8BEC            MOV     BP,SP
                                        ; STATEMENT # 292
0251  FF1E0000        CALL    UIES
                                        ; STATEMENT # 294
0255  5D              POP     BP
0256  C3              RET
            P_UIM       ENDP
                                        ; STATEMENT # 383
```

```
 69                    CSR_OFF. . . . . LITERALLY '01$00000B'
157  0000H            CURSOR . . . . . PROCEDURE EXTERNAL(46) STACK=0000H       281
  2                    DC . . . . . . . LITERALLY 'DECLARE'                3     4     5     6     7     8     9
                                          10    11    12    13    14    15    16    17    18    19    39    40
                                          41    42    43    44    45    46    47    48    49    50    51    52
                                          53    54    55    56    57    58    59    60    61    62    63    64
                                          65    66    67    68    69    70    71    72    73    74    75    76
                                          77    78    79    80    81    82    83    84    85    86    87    88
                                          89    90    91    92    93    94    95    96    97    98    99   100
                                         101   102   103   104   105   106   107   108   109   110   111   112
                                         113   114   115   116   117   118   119   120   121   122   123   124
                                         125   126   127   128   129   130   131   132   133   134   135   136
                                         137   138   139   140   141   142   143   144   145   146   147   148
                                         185   186   187   188
244  0138H     67  DCA. . . . . . . . PROCEDURE PUBLIC STACK=0006H
 79  0000H      4  DC1. . . . . . . . POINTER EXTERNAL(11)
 11                DEL. . . . . . . . LITERALLY '07FH'       214  215
 80  0000H      4  DFC. . . . . . . . POINTER EXTERNAL(12)
 95  0000H    256  DMAP . . . . . . . BYTE ARRAY(256) EXTERNAL(27)
255  017BH     50  DOC. . . . . . . . PROCEDURE BYTE STACK=0004H         224
 77  0000H      2  DS_SIZE. . . . . . WORD EXTERNAL(9)
 21  0000H        DXMTC. . . . . . . PROCEDURE EXTERNAL(0) STACK=0000H
159  0000H        D_CRT. . . . . . . PROCEDURE EXTERNAL(47) STACK=0000H       215  216
162  0000H        D_TTY_RES. . . . . PROCEDURE EXTERNAL(48) STACK=0000H
 81  0000H      4  D_XMTC . . . . . . POINTER EXTERNAL(13)
 82  0000H      4  EDC. . . . . . . . POINTER EXTERNAL(14)        287
284  0239H     21  EDL. . . . . . . . PROCEDURE PUBLIC STACK=000EH
 83  0000H      4  EMEC . . . . . . . POINTER EXTERNAL(15)
168  0000H        ENABLE_LINES . . . PROCEDURE EXTERNAL(51) STACK=0000H
 12                EOL. . . . . . . . LITERALLY '080H'
 13                EOS. . . . . . . . LITERALLY '030H'
 14                ESC. . . . . . . . LITERALLY '01BH'       212
109  0000H     10  ESCP . . . . . . . STRUCTURE EXTERNAL(38)
      0000H      1      COMMAND. . . . BYTE       223*  237*  269  270  271  272  273  275  276  277
      0001H      2      FUNCTION . . . WORD       270*  272*  274*  276*  278*  279*  280
      0003H      1      MODE . . . . . BYTE       198*  210  219*  220*  226*  268*
      0004H      1      OPER_COUNT . . BYTE       224*  227  233
      0005H      1      OPER_INDEX . . BYTE       225*  232*  232  233
      0006H      4      OPERAND. . . . BYTE ARRAY(4)            231*
173  0000H     40  ESC_TABLE_1. . . . WORD ARRAY(20) DATA          270
194  0028H     14  ESC_TABLE_2. . . . WORD ARRAY(7) DATA      272
195  0036H     42  ESC_TABLE_3. . . . WORD ARRAY(21) DATA          276
  5                FALSE. . . . . . . LITERALLY '000H'       200
 15                FF . . . . . . . . LITERALLY '00CH'
 24  0000H        FLAGS. . . . . . . PROCEDURE WORD EXTERNAL(1) STACK=0000H
 84  0000H      4  FONT . . . . . . . POINTER EXTERNAL(16)
 93  0000H      2  FONTSIZE . . . . . WORD EXTERNAL(25)
172  0000H      1  FORE . . . . . . . BYTE IN PROC (SCA) PARAMETER       172
      006EH        H19CRT . . . . . . PROCEDURE STACK=0000H
110  0000H     18  H19_MODE . . . . . STRUCTURE EXTERNAL(39)
      0000H      1      ALTERNATE. . . BYTE       200*
      0001H      1      ANSI . . . . . BYTE       200*  218
      0002H      1      AUTO_CR. . . . BYTE
      0003H      1      AUTO_LF. . . . BYTE
      0004H      1      AUTO_REPEAT. . BYTE
```

```
135                 KY_F12 . . . . . .   LITERALLY '0A2H'
133                 KY_HELP. . . . . .   LITERALLY '095H'
142                 KY_HOME. . . . . .   LITERALLY '0A9H'
136                 KY_ID_CHAR . . . .   LITERALLY '0A3H'
137                 KY_ID_LINE . . . .   LITERALLY '0A4H'
146                 KY_KP_0. . . . . .   LITERALLY '0B0H'
147                 KY_KP_9. . . . . .   LITERALLY '0B9H'
144                 KY_KP_MINUS. . . .   LITERALLY '0ADH'
145                 KY_KP_PERIOD . . .   LITERALLY '0AEH'
141                 KY_LEFT. . . . . .   LITERALLY '0A8H'
140                 KY_RIGHT . . . . .   LITERALLY '0A7H'
148                 KY_SHIFT_OFFSET. .   LITERALLY '040H'
138                 KY_UP. . . . . . .   LITERALLY '0A5H'
 17                 LF . . . . . . . .   LITERALLY '00AH'
285  0004H      1  LINE . . . . . . .   BYTE IN PROC (EDL) PARAMETER AUTOMATIC          285  287
169  0000H      1  LINES. . . . . . .   BYTE IN PROC (ENABLE_LINES) PARAMETER           169
190  0000H      1  LINE_INDEX . . . .   BYTE PUBLIC
  3                 LT . . . . . . . .   LITERALLY 'LITERALLY'              4    5    6    7    8    9   10
                                           11   12   13   14   15   16   17   18   19   20   40   41
                                           42   43   44   45   46   47   48   49   50   51   52   53
                                           54   55   56   57   58   59   60   61   62   63   64   65
                                           66   67   68   69   70   71   72   73   74  103  104  105
                                          113  114  115  116  117  118  119  120  121  122  123  124
                                          125  126  127  128  129  130  131  132  133  134  135  136
                                          137  138  139  140  141  142  143  144  145  146  147  148
                                          186  187  188  189
177  0000H      1  MASK . . . . . . .   BYTE IN PROC (SCP1) PARAMETER       177
 85  0000H      4  MDC. . . . . . . .   POINTER EXTERNAL(17)
 86  0000H      4  MDL. . . . . . . .   POINTER EXTERNAL(18)
 74                 MTR_ISSUE. . . . .   LITERALLY '54H'
 73                 MTR_VERSION. . . .   LITERALLY '12H'
189                 M_ANS. . . . . . .   LITERALLY '3'            219
187                 M_ESC. . . . . . .   LITERALLY '1'            220
186                 M_NRM. . . . . . .   LITERALLY '0'            198  268
188                 M_OPR. . . . . . .   LITERALLY '2'            226
254  0060H     14  OPC. . . . . . . .   BYTE ARRAY(14) DATA             260  263
266  01ADH    140  PES. . . . . . . .   PROCEDURE STACK=0004H           228  234
 87  0000H      4  PROMPT . . . . . .   POINTER EXTERNAL(19)
100  0000H      1  PSP. . . . . . . .   BYTE EXTERNAL(32)
295  0000H         P_CDN. . . . . . .   PROCEDURE EXTERNAL(58) STACK=0000H          193
297  0000H         P_CLF. . . . . . .   PROCEDURE EXTERNAL(59) STACK=0000H          193
369  0000H         P_COLOR. . . . . .   PROCEDURE EXTERNAL(95) STACK=0000H          195
303  0000H         P_CPR. . . . . . .   PROCEDURE EXTERNAL(62) STACK=0000H          195
299  0000H         P_CRT. . . . . . .   PROCEDURE EXTERNAL(60) STACK=0000H          193
301  0000H         P_CUP. . . . . . .   PROCEDURE EXTERNAL(61) STACK=0000H          193
327  0000H         P_DC . . . . . . .   PROCEDURE EXTERNAL(74) STACK=0000H          193
305  0000H         P_DCA. . . . . . .   PROCEDURE EXTERNAL(63) STACK=0000H          194
347  0000H         P_DK . . . . . . .   PROCEDURE EXTERNAL(84) STACK=0000H          195
329  0000H         P_DL . . . . . . .   PROCEDURE EXTERNAL(75) STACK=0000H          193
333  0000H         P_EAKM . . . . . .   PROCEDURE EXTERNAL(77) STACK=0000H          193
337  0000H         P_EAM. . . . . . .   PROCEDURE EXTERNAL(79) STACK=0000H          193
315  0000H         P_EBOL . . . . . .   PROCEDURE EXTERNAL(68) STACK=0000H          195
317  0000H         P_EBOP . . . . . .   PROCEDURE EXTERNAL(69) STACK=0000H          274
319  0000H         P_EEOL . . . . . .   PROCEDURE EXTERNAL(70) STACK=0000H          193
321  0000H         P_EEOP . . . . . .   PROCEDURE EXTERNAL(71) STACK=0000H          193
343  0000H         P_EGM. . . . . . .   PROCEDURE EXTERNAL(82) STACK=0000H          193
```

```
  97  0000H     1  VERT_LINE. . . . .    BYTE EXTERNAL(29)       248*
  57               VIDC_BLUE. . . . .    LITERALLY '00000100B'
  55               VIDC_FLSH. . . . .    LITERALLY '00001000B'
  58               VIDC_GRNE. . . . .    LITERALLY '00000010B'
  60               VIDC_RAME. . . . .    LITERALLY '10000000B'
  59               VIDC_REDE. . . . .    LITERALLY '00000001B'
  52               VIDC_WBLU. . . . .    LITERALLY '01000000B'
  53               VIDC_WGRN. . . . .    LITERALLY '00100000B'
  54               VIDC_WRED. . . . .    LITERALLY '00010000B'
  51               VIDC_WXXX. . . . .    LITERALLY '01110000B'
  56               VIDC_XXXE. . . . .    LITERALLY '00000111B'
  32  0000H        VIDEO_INTR_A . . .    PROCEDURE EXTERNAL(4) STACK=0000H
  44               VID_CDC. . . . . .    LITERALLY 'IO_VIDEO+1'
  43               VID_CDD. . . . . .    LITERALLY 'IO_VIDEO+0'
  42               VID_CMD. . . . . .    LITERALLY 'IO_VIDEO+0'
  45               VRMM_DAT . . . . .    LITERALLY 'IO_VIDEO+2'
  47               VRMM_MPC . . . . .    LITERALLY 'IO_VIDEO+3'
  46               VRMM_MPD . . . . .    LITERALLY 'IO_VIDEO+2'
  18               VT . . . . . . . .    LITERALLY '00BH'
  75  0000H     5  WIP. . . . . . . .    BYTE ARRAY(5) EXTERNAL(7)
 154  0000H        WRITK. . . . . . .    PROCEDURE EXTERNAL(45) STACK=0000H
  91  0000H     4  XCA. . . . . . . .    POINTER EXTERNAL(23)
 105               XCOLOR_BACK. . . .    LITERALLY '00$111$000B'
 104               XCOLOR_FORE. . . .    LITERALLY '00$000$111B'
  34  0000H        XEC. . . . . . . .    PROCEDURE EXTERNAL(5) STACK=0000H
 112  0000H    10  XMT. . . . . . . .    STRUCTURE EXTERNAL(41)
      0000H     1      BURST. . . . .    BYTE
      0001H     2      BCOUNT . . . .    INTEGER
      0003H     2      COUNT. . . . .    WORD       207
      0005H     1      COL. . . . . .    BYTE
      0006H     1      ROW. . . . . .    BYTE
      0007H     1      COLOR. . . . .    BYTE
      0008H     1      GRAPHIC. . . .    BYTE
      0009H     1      REVERSE. . . .    BYTE
 183  0000H        XMTSC. . . . . . .    PROCEDURE EXTERNAL(57) STACK=0000H
  19               XOFF . . . . . . .    LITERALLY '013H'
  20               XON. . . . . . . .    LITERALLY '011H'
 103               XREVERSE . . . . .    LITERALLY '10$000$000B'


MODULE INFORMATION:

     CODE AREA SIZE     = 0257H    599D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0004H      4D
     MAXIMUM STACK SIZE = 000EH     14D
     2134 LINES READ
     0 PROGRAM WARNINGS
     0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION
```

```
SERIES-III 8086/8087/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE H19_CRT_A86
OBJECT MODULE PLACED IN :F2:H19A.OBJ
INVOCATION LINE CONTROLS:  PRINT(:TO:) XREF ERRORPRINT(:TO:)


LOC  OBJ                  LINE     SOURCE

                             1     ;;      h19crt.a86
                             2     ;
                             3     ;       These routines are optimized PL/M-86 routines.
                             4     ;       The output of the PL/M-86 compiler has been
                             5     ;       extensively 'filtered' to produce files which
                             6     ;       contain optimized code.  The basic optimization
                             7     ;       was to eliminate the procedure prologue/epilogue
                             8     ;       code for procedures which pass no parameters.
                             9     ;       This consists of reducing the following:
                            10     ;
                            11     ;               xxxx    PROC    NEAR
                            12     ;                       PUSH    BP
                            13     ;                       MOV     BP,SP
                            14     ;                         :       :
                            15     ;                         :       :
                            16     ;                       POP     BP
                            17     ;                       RET
                            18     ;               xxxx    ENDP
                            19     ;
                            20     ;       to:
                            21     ;
                            22     ;               xxxx    PROC    NEAR
                            23     ;                         :       :
                            24     ;                         :       :
                            25     ;                       RET
                            26     ;               xxxx    ENDP
                            27     ;
                            28     ;       This saves at least four bytes of code for each
                            29     ;       of the procedures involved.
                            30     ;
                            31
                            32 +1  $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

0003                  =2     87          AUTO_LF      DB      ?       ; Auto Line-Feed on Carriage-Return
0004                  =2     88          AUTO_REPEAT  DB      ?       ; Auto-Repeat Keyboard
0005                  =2     89          BWO          DB      ?       ; Black and White Optimization
0006                  =2     90          CURSOR       DB      ?       ; Programmed Cursor Value
0007                  =2     91          CURSOR_ON    DB      ?       ; Cursor Enabled
0008                  =2     92          EXPAND       DB      ?       ; Expand Key-Board Characters
0009                  =2     93          GRAPHIC      DB      ?       ; Graphic Character Mode
000A                  =2     94          INSERT       DB      ?       ; Insert Character Mode
000B                  =2     95          KEY_EN       DB      ?       ; Key-Board Enable
000C                  =2     96          REVERSE      DW      ?       ; Reverse Video
000E                  =2     97          SHIFTED      DB      ?       ; Shifted Key-Pad Mode
000F                  =2     98          STATUS       DB      ?       ; Status-Line Enabled
0010                  =2     99          UPDN         DB      ?       ; Key-Board Up/Down Mode
0011                  =2    100          WRAP         DB      ?       ; Wrap at End-of-Line
------                =2    101        H19_MODE_STRUC           ENDS
                      =2    102
----                  =2    103      H19_PAR_STRUC    STRUC
0000                  =2    104          CPL          DB      ?       ; Characters Per Line
0001                  =2    105          DSC          DB      ?       ; Displayed Scan Lines per Character
0002                  =2    106          LPS          DB      ?       ; Lines Per Screen
0003                  =2    107          POVRAM       DB      ?       ; Planes of Video RAM
0004                  =2    108          SLI          DB      ?       ; Status Line Index
0005                  =2    109          SPC          DB      ?       ; Scan Lines per Character
0006                  =2    110          SW401        DB      ?       ; H19-Switch 401
0007                  =2    111          SW402        DB      ?       ; H19-Switch 402
0008                  =2    112          VRAM_SIZE    DB      ?       ; 0-32KBytes, 1-64KBytes
------                =2    113        H19_PAR_STRUC ENDS
                      =2    114
------                =2    115      XMT_STRUC        STRUC
0000                  =2    116          BURST        DB      ?       ; Characters to Transmit per VSYNC
0001                  =2    117          BCOUNT       DW      ?       ; Remaining Characters in current Burst
0003                  =2    118          COUNT        DW      ?       ; Characters left to Transmit
0005                  =2    119          COL          DB      ?       ; Horizontal Column to Transmit
0006                  =2    120          ROW          DB      ?       ; Vertical   Row    to Transmit
0007                  =2    121          XMT_COLOR    DB      ?       ; Current COLOR   State Transmitted
0008                  =2    122          XMT_GRAPHIC  DB      ?       ; Current GRAPHIC State Transmitted
0009                  =2    123          XMT_REVERSE  DB      ?       ; Current REVERSE State Transmitted
----                  =2    124        XMT_STRUC      ENDS
                      =2    125
                      =2    126
                      =2    127      $RESTORE
                      =1    128
                      =1    129      ;       Globals Definitions
                      =1    130
                      =1    131      %IF(%NES(ASTLIB,%SEGMENT_LABEL)) THEN (
                      =1               ASTLIB         SEGMENT BYTE PUBLIC 'CODE'
                      =1                              EXTRN   VIDEO_INTR_A:FAR
                      =1               ASTLIB         ENDS
                      =1               )FI
-----                 =1    136
                      =1    137
                      =1    138      %IF(%NES(FONTAB,%SEGMENT_LABEL)) THEN (
                      =1               FONTAB         SEGMENT BYTE PUBLIC 'DATA'
                      =1                              EXTRN   MTR_FONT:BYTE
                      =1               FONTAB         ENDS
```

```
LOC  OBJ                  LINE     SOURCE

                          =1                              EXTRN   DFC:DWORD          ; Display Font Character
                          =1                              EXTRN   D_XMTC:DWORD       ; Dumb Terminal Transmit Character
                          =1                              EXTRN   EDC:DWORD          ; Erase Display Character
                          =1                              EXTRN   EMEC:DWORD         ; Extend-Mode Escape Character
                          =1                              EXTRN   FONT:DWORD         ; Pointer to Font Table
                          =1                              EXTRN   MDC:DWORD          ; Move Display Character
                          =1                              EXTRN   MDL:DWORD          ; Move Display Line
                          =1                              EXTRN   PROMPT:DWORD       ; Display Monitor Prompt
                          =1                              EXTRN   RDC:DWORD          ; Read Display Character
                          =1                              EXTRN   S_XMTC:DWORD       ; Smart Terminal Transmit Character
                          =1                              EXTRN   UIES:DWORD         ; Un-Implemented Escape Sequence
                          =1                              EXTRN   XCA:DWORD          ; Transmit Character Attributes
                          =1
                          =1                              EXTRN   BOOT_PAR:BOOT_PAR_STRUC
                          =1                              EXTRN   COLOR:COLOR_STRUC
                          =1                              EXTRN   CRTC_CURSOR:CRTC_STRUC
                          =1                              EXTRN   CRTC_DISPLAY:CRTC_STRUC
                          =1                              EXTRN   ESCP:ESCP_STRUC
                          =1                              EXTRN   H19_MODE:H19_MODE_STRUC
                          =1                              EXTRN   H19_PAR:H19_PAR_STRUC
                          =1                              EXTRN   XMT:XMT_STRUC
                          =1                DATA          ENDS
                          =1                )FI
                          =1    218
                          =1    219
                          =1    220         %IF(%NES(VIDEOA,%SEGMENT_LABEL)) THEN (
                          =1                  VIDEOA      SEGMENT BYTE PUBLIC 'CODE'
                          =1                  VIDEOA      ENDS
                          =1                  )FI
                          =1    224
                          =1    225
                          =1    226         CGROUP  GROUP   MTR100,CODE,ASTLIB,VIDEOA
                          =1    227
                          =1    228
                          =1    229         $RESTORE
                                230
                                231 +1      $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           287     P_CDN       ENDP
                           288                                 ; STATEMENT # 301
0018                       289     P_CLF       PROC NEAR
                           290             PUBLIC  P_CLF
                           291     ;       PUSH    BP
                           292     ;       MOV     BP,SP
                           293                                 ; STATEMENT # 303
0018 A00000          E     294             MOV     AL,HORZ_CHAR
001B 0AC0                  295             OR      AL,AL
001D 740A                  296             JZ      @34
                           297                                 ; STATEMENT # 304
001F FF360000        E     298             PUSH    WORD PTR VERT_LINE; 1
0023 FEC8                  299             DEC     AL
0025 50                    300             PUSH    AX      ; 2
0026 E80000          E     301             CALL    DCA
0029                       302     @34:
                           303                                  ; STATEMENT # 306
                           304     ;       POP     BP
0029 C3                    305             RET
                           306     P_CLF       ENDP
                           307                                 ; STATEMENT # 307
002A                       308     P_CRT       PROC NEAR
                           309             PUBLIC  P_CRT
                           310     ;       PUSH    BP
                           311     ;       MOV     BP,SP
                           312                                 ; STATEMENT # 309
002A FF360000        E     313             PUSH    WORD PTR VERT_LINE; 1
002E A00000          E     314             MOV     AL,HORZ_CHAR
0031 FEC0                  315             INC     AL
0033 50                    316             PUSH    AX      ; 2
0034 E80000          E     317             CALL    DCA
                           318                                 ; STATEMENT # 311
                           319     ;       POP     BP
0037 C3                    320             RET
                           321     P_CRT       ENDP
                           322                                 ; STATEMENT # 312
0038                       323     P_CUP       PROC NEAR
                           324             PUBLIC  P_CUP
                           325     ;       PUSH    BP
                           326     ;       MOV     BP,SP
                           327                                 ; STATEMENT # 314
0038 A00000          E     328             MOV     AL,VERT_LINE
003B 3A060400        E     329             CMP     AL,H19_PAR.SLI
003F 730A                  330             JNB     @35
                           331                                 ; STATEMENT # 315
0041 FEC8                  332             DEC     AL
0043 50                    333             PUSH    AX      ; 1
0044 FF360000        E     334             PUSH    WORD PTR HORZ_CHAR; 2
0048 E80000          E     335             CALL    DCA
004B                       336     @35:
                           337                                 ; STATEMENT # 317
                           338     ;       POP     BP
004B C3                    339             RET
                           340     P_CUP       ENDP
                           341                                 ; STATEMENT # 318
```

```
LOC  OBJ                  LINE     SOURCE

                           397     P_HOM       ENDP
                           398                                 ; STATEMENT # 336
0082                       399     P_RCP       PROC NEAR
                           400             PUBLIC  P_RCP
                           401     ;       PUSH    BP
                           402     ;       MOV     BP,SP
                           403                                 ; STATEMENT # 338
0082 FF360000         E    404             PUSH    WORD PTR SAVED_VERT_LINE; 1
0086 FF360000         E    405             PUSH    WORD PTR SAVED_HORZ_CHAR; 2
008A E80000           E    406             CALL    DCA
                           407                                 ; STATEMENT # 340
                           408     ;       POP     BP
008D C3                    409             RET
                           410     P_RCP       ENDP
                           411                                 ; STATEMENT # 341
008E                       412     P_RLF       PROC NEAR
                           413             PUBLIC  P_RLF
                           414     ;       PUSH    BP
                           415     ;       MOV     BP,SP
                           416                                 ; STATEMENT # 343
008E A00000           E    417             MOV     AL,VERT_LINE
0091 3A060400         E    418             CMP     AL,H19_PAR.SLI
0095 7501                  419             JNZ     @36
                           420                                 ; STATEMENT # 344
                           421     ;       POP     BP
0097 C3                    422              RET
                           423                                 ; STATEMENT # 345
0098                       424     @36:
0098 803E000000       E    425             CMP     VERT_LINE,0H
009D 7405                  426             JZ      @38
                           427                                 ; STATEMENT # 346
009F FE0E0000         E    428             DEC     VERT_LINE
                           429                                 ; STATEMENT # 347
                           430     ;       POP     BP
00A3 C3                    431             RET
00A4                       432     @38:
00A4 E80000           E    433             CALL    REV_SCROLL
                           434                                 ; STATEMENT # 349
                           435     ;       POP     BP
00A7 C3                    436             RET
                           437     P_RLF       ENDP
                           438                                 ; STATEMENT # 350
00A8                       439     P_SCP       PROC NEAR
                           440             PUBLIC  P_SCP
                           441     ;       PUSH    BP
                           442     ;       MOV     BP,SP
                           443                                 ; STATEMENT # 352
00A8 A00000           E    444             MOV     AL,HORZ_CHAR
00AB A20000           E    445             MOV     SAVED_HORZ_CHAR,AL
                           446                                 ; STATEMENT # 353
00AE A00000           E    447             MOV     AL,VERT_LINE
00B1 A20000           E    448             MOV     SAVED_VERT_LINE,AL
                           449                                 ; STATEMENT # 355
                           450     ;       POP     BP
00B4 C3                    451             RET
```

```
LOC  OBJ                  LINE     SOURCE

                           507     ;       PUSH    BP
                           508     ;       MOV     BP,SP
                           509                             ; STATEMENT # 374
00F7 FF360000        E     510             PUSH    WORD PTR VERT_LINE; 1
00FB A00000          E     511             MOV     AL,HORZ_CHAR
00FE 50                    512             PUSH    AX      ; 2
00FF A00000          E     513             MOV      AL,H19_PAR.CPL
0102 2A060000        E     514             SUB     AL,HORZ_CHAR
0106 50                    515             PUSH    AX      ; 3
0107 FF1E0000        E     516             CALL    EDC
                           517                             ; STATEMENT # 376
                           518     ;       POP     BP
010B C3                    519             RET
                           520     P_EEOL      ENDP
                           521                             ; STATEMENT # 377
010C                       522     P_EEOP      PROC NEAR
                           523             PUBLIC  P_EEOP
                           524     ;       PUSH    BP
                           525     ;       MOV     BP,SP
                           526                             ; STATEMENT # 379
010C E8E8FF                527             CALL    P_EEOL
                           528                             ; STATEMENT # 380
010F A00000          E     529             MOV     AL,VERT_LINE
0112 FEC0                  530             INC     AL
0114 A20000          E     531             MOV     LINE_INDEX,AL
0117                       532     @43:
0117 A00400          E     533             MOV     AL,H19_PAR.SLI
011A FEC8                  534             DEC     AL
011C 8A0E0000        E     535             MOV     CL,LINE_INDEX
0120 3AC8                  536             CMP     CL,AL
0122 770A                  537             JA      @44
                           538                             ; STATEMENT # 381
0124 51                    539              PUSH    CX      ; 1
0125 E80000          E     540             CALL    EDL
                           541                             ; STATEMENT # 382
0128 FE060000        E     542             INC     LINE_INDEX
012C 75E9                  543             JNZ     @43
012E                       544     @44:
                           545                             ; STATEMENT # 384
                           546     ;       POP     BP
012E C3                    547             RET
                           548     P_EEOP      ENDP
                           549                             ; STATEMENT # 385
012F                       550     P_ELIN      PROC NEAR
                           551             PUBLIC  P_ELIN
                           552     ;       PUSH    BP
                           553     ;       MOV     BP,SP
                           554                             ; STATEMENT # 387
012F FF360000        E     555             PUSH    WORD PTR VERT_LINE; 1
0133 E80000          E     556             CALL    EDL
                           557                             ; STATEMENT # 389
                           558     ;       POP     BP
0136 C3                    559             RET
                           560     P_ELIN      ENDP
                           561                             ; STATEMENT # 390
```

```
LOC  OBJ                  LINE     SOURCE

0181 B400                  617              MOV     AH,0H
0183 50                    618              PUSH    AX      ; 2
0184 A00000          E     619              MOV     AL,HORZ_CHAR
0187 50                    620              PUSH    AX      ; 3
0188 A00000          E     621              MOV      AL,H19_PAR.CPL
018B 2A060000        E     622              SUB     AL,HORZ_CHAR
018F FEC8                  623              DEC     AL
0191 B400                  624              MOV     AH,0H
0193 50                    625              PUSH    AX      ; 4
0194 FF1E0000        E     626              CALL    MDC
                           627                              ; STATEMENT # 407
0198 FF360000        E     628              PUSH    WORD PTR VERT_LINE; 1
019C A00000          E     629              MOV      AL,H19_PAR.CPL
019F FEC8                  630              DEC     AL
01A1 B400                  631              MOV     AH,0H
01A3 50                    632              PUSH    AX      ; 2
01A4 B001                  633              MOV     AL,1H
01A6 50                    634              PUSH    AX      ; 3
01A7 FF1E0000        E     635              CALL    EDC
                           636                              ; STATEMENT # 409
                           637      ;       POP     BP
01AB C3                    638              RET
                           639      P_DC      ENDP
                           640                              ; STATEMENT # 410
01AC                       641      P_DL      PROC NEAR
                           642              PUBLIC  P_DL
                           643      ;       PUSH    BP
                           644      ;       MOV     BP,SP
                           645                              ; STATEMENT # 412
01AC A00000          E     646              MOV     AL,VERT_LINE
01AF 3A060400        E     647              CMP     AL,H19_PAR.SLI
01B3 742E                  648              JZ      @50
                           649                              ; STATEMENT # 414
01B5 FEC0                  650              INC     AL
01B7 A20000          E     651              MOV     LINE_INDEX,AL
01BA                       652      @51:
01BA A00400          E     653              MOV     AL,H19_PAR.SLI
01BD FEC8                  654              DEC     AL
01BF 8A0E0000        E     655               MOV     CL,LINE_INDEX
01C3 3AC8                  656              CMP     CL,AL
01C5 7710                  657              JA      @52
                           658                              ; STATEMENT # 415
01C7 51                    659              PUSH    CX      ; 1
01C8 FEC9                  660              DEC     CL
01CA B500                  661              MOV     CH,0H
01CC 51                    662              PUSH    CX      ; 2
01CD FF1E0000        E     663              CALL    MDL
                           664                              ; STATEMENT # 416
01D1 FE060000        E     665              INC     LINE_INDEX
01D5 75E3                  666              JNZ     @51
01D7                       667      @52:
                           668                              ; STATEMENT # 417
01D7 A00400          E     669              MOV     AL,H19_PAR.SLI
01DA FEC8                  670              DEC     AL
01DC 50                    671              PUSH    AX      ; 1
```

```
LOC  OBJ                  LINE     SOURCE

0229                       727     P_EAKM      PROC NEAR
                           728             PUBLIC  P_EAKM
                           729     ;       PUSH    BP
                           730     ;       MOV     BP,SP
                           731                             ; STATEMENT # 439
0229 C6060000FF      E     732             MOV     H19_MODE.ALTERNATE,0FFH
                           733                             ; STATEMENT # 441
                           734     ;       POP     BP
022E C3                    735             RET
                           736     P_EAKM      ENDP
                           737                             ; STATEMENT # 442
022F                       738     P_XAKM      PROC NEAR
                           739             PUBLIC  P_XAKM
                           740     ;       PUSH    BP
                           741     ;       MOV     BP,SP
                           742                             ; STATEMENT # 444
022F C606000000      E     743             MOV     H19_MODE.ALTERNATE,0H
                           744                             ; STATEMENT # 446
                           745     ;       POP     BP
0234 C3                    746             RET
                           747     P_XAKM      ENDP
                           748                             ; STATEMENT # 447
0235                       749     P_EAM       PROC NEAR
                           750             PUBLIC  P_EAM
                           751     ;       PUSH    BP
                           752     ;       MOV     BP,SP
                           753                             ; STATEMENT # 449
0235 C6060100FF      E     754             MOV     H19_MODE.ANSI,0FFH
                           755                             ; STATEMENT # 451
                           756     ;       POP     BP
023A C3                    757             RET
                           758     P_EAM       ENDP
                           759                             ; STATEMENT # 452
023B                       760     P_EICM      PROC NEAR
                           761             PUBLIC  P_EICM
                           762     ;       PUSH    BP
                           763     ;       MOV     BP,SP
                           764                             ; STATEMENT # 454
023B C6060A00FF      E     765             MOV     H19_MODE.INSERT,0FFH
                           766                             ; STATEMENT # 456
                           767     ;       POP     BP
0240 C3                    768             RET
                           769     P_EICM      ENDP
                           770                             ; STATEMENT # 457
0241                       771     P_XICM      PROC NEAR
                           772             PUBLIC  P_XICM
                           773     ;       PUSH    BP
                           774     ;       MOV     BP,SP
                           775                             ; STATEMENT # 459
0241 C6060A0000      E     776             MOV     H19_MODE.INSERT,0H
                           777                             ; STATEMENT # 461
                           778     ;       POP     BP
0246 C3                    779             RET
                           780     P_XICM      ENDP
                           781                             ; STATEMENT # 462
```

```
LOC  OBJ                  LINE     SOURCE

                           837     ;       MOV     BP,SP
                           838                             ; STATEMENT # 486
026B C6060E00FF      E     839             MOV     H19_MODE.SHIFTED,0FFH
                           840                             ; STATEMENT # 488
                           841     ;       POP     BP
0270 C3                    842             RET
                           843     P_EKPS      ENDP
                           844                             ; STATEMENT # 489
0271                       845     P_XKPS      PROC NEAR
                           846             PUBLIC  P_XKPS
                           847     ;       PUSH    BP
                           848     ;       MOV     BP,SP
                           849                             ; STATEMENT # 491
0271 C6060E0000      E     850             MOV     H19_MODE.SHIFTED,0H
                           851                             ; STATEMENT # 493
                           852     ;       POP     BP
0276 C3                    853             RET
                           854     P_XKPS      ENDP
                           855                             ; STATEMENT # 494
0277                       856     P_ERVM      PROC NEAR
                           857             PUBLIC  P_ERVM
                           858     ;       PUSH    BP
                           859     ;       MOV     BP,SP
                           860                             ; STATEMENT # 496
0277 C7060C00FFFF    E     861             MOV     H19_MODE.REVERSE,0FFFFH
                           862                             ; STATEMENT # 498
                           863     ;       POP     BP
027D C3                    864             RET
                           865     P_ERVM      ENDP
                           866                             ; STATEMENT # 499
027E                       867     P_XRVM      PROC NEAR
                           868             PUBLIC  P_XRVM
                           869     ;       PUSH    BP
                           870     ;       MOV     BP,SP
                           871                             ; STATEMENT # 501
027E C7060C000000    E     872             MOV     H19_MODE.REVERSE,0H
                           873                             ; STATEMENT # 503
                           874     ;       POP     BP
0284 C3                    875             RET
                           876     P_XRVM      ENDP
                           877                             ; STATEMENT # 504
0285                       878     P_EWRAP     PROC NEAR
                           879             PUBLIC  P_EWRAP
                           880     ;       PUSH    BP
                           881     ;       MOV     BP,SP
                           882                             ; STATEMENT # 506
0285 C6061100FF      E     883             MOV     H19_MODE.WRAP,0FFH
                           884                             ; STATEMENT # 508
                           885     ;       POP     BP
028A C3                    886             RET
                           887     P_EWRAP     ENDP
                           888                             ; STATEMENT # 509
028B                       889     P_XWRAP     PROC NEAR
                           890             PUBLIC  P_XWRAP
                           891     ;       PUSH    BP
```

```
LOC  OBJ                  LINE     SOURCE

02D1 2903           R      947              DW      CGROUP:@74
02D3 2F03           R      948              DW      CGROUP:@75
02D5                       949      @60:
                           950                                       ; STATEMENT # 526
02D5 C6060F0000     E      951              MOV     H19_MODE.STATUS,0H
                           952                                       ; STATEMENT # 527
02DA FF360400       E      953               PUSH    WORD PTR H19_PAR.SLI; 1
02DE E80000         E      954              CALL    EDL
                           955                                       ; STATEMENT # 528
02E1 FF360400       E      956              PUSH    WORD PTR H19_PAR.SLI; 1
02E5 E80000         E      957              CALL    ENABLE_LINES
                           958                                       ; STATEMENT # 530
                           959      ;       POP     BP
02E8 C3                    960              RET
02E9                       961      @61:
02E9 B003                  962              MOV     AL,3H
02EB EB4990                963              JMP     @30
                           964                                       ; STATEMENT # 532
02EE                       965      @63:
02EE B0E0                  966              MOV     AL,0E0H
02F0 50                    967              PUSH    AX      ; 1
02F1 A00500         E      968              MOV     AL,H19_PAR.SPC
02F4 2C03                  969              SUB     AL,3H
02F6 EB2390                970              JMP     @26
02F9                       971      @64:
                           972                                       ; STATEMENT # 534
02F9 C6060700FF     E      973              MOV     H19_MODE.CURSOR_ON,0FFH
                           974                                       ; STATEMENT # 535
02FE E80000         E      975              CALL    SCP
                           976                                       ; STATEMENT # 537
                           977      ;       POP     BP
0301 C3                    978              RET
0302                       979      @65:
0302 E86CFF                980              CALL    P_XKPS
                           981                                       ; STATEMENT # 538
                           982      ;       POP     BP
0305 C3                    983              RET
0306                       984      @66:
0306 E826FF                985              CALL    P_XAKM
                           986                                       ; STATEMENT # 539
                           987      ;       POP     BP
0309 C3                    988              RET
030A                       989      @67:
030A C606030000     E      990              MOV     H19_MODE.AUTO_LF,0H
                           991                                       ; STATEMENT # 540
                           992      ;       POP     BP
030F C3                    993              RET
0310                       994      @68:
0310 C606020000     E      995              MOV     H19_MODE.AUTO_CR,0H
                           996                                       ; STATEMENT # 541
                           997      ;       POP     BP
0315 C3                    998              RET
                           999                                       ; STATEMENT # 542
0316                      1000      @70:
0316 B01F                 1001              MOV     AL,1FH
```

```
LOC  OBJ                  LINE     SOURCE

0358                      1057     @78:
0358 7803           R     1058     LB2       DW      CGROUP:@79
035A 9503           R     1059               DW      CGROUP:@81
035C CB03           R     1060               DW      CGROUP:@49
035E 9A03           R     1061               DW      CGROUP:@83
0360 9F03           R     1062               DW      CGROUP:@84
0362 A803           R     1063               DW      CGROUP:@85
0364 AC03           R     1064               DW      CGROUP:@86
0366 B003           R     1065               DW      CGROUP:@87
0368 B603           R     1066               DW      CGROUP:@88
036A CB03           R     1067               DW      CGROUP:@49
036C BC03           R     1068               DW      CGROUP:@90
036E C603           R     1069               DW      CGROUP:@91
0370 CB03           R     1070               DW      CGROUP:@49
0372 CB03           R     1071               DW      CGROUP:@49
0374 CF03           R     1072               DW      CGROUP:@94
0376 D503           R     1073               DW      CGROUP:@95
0378                      1074     @79:
                          1075                       ; STATEMENT # 559
0378 A00F00         E     1076               MOV     AL,H19_MODE.STATUS
037B D0D8                 1077               RCR     AL,1
037D 7261                 1078               JB      @76
                          1079                       ; STATEMENT # 561
037F C6060F00FF     E     1080               MOV     H19_MODE.STATUS,0FFH
                          1081                       ; STATEMENT # 562
0384 FF360400       E     1082               PUSH    WORD PTR H19_PAR.SLI; 1
0388 E80000         E     1083               CALL    EDL
                          1084                       ; STATEMENT # 563
038B A00400         E     1085               MOV     AL,H19_PAR.SLI
038E FEC0                 1086               INC     AL
0390 50                   1087               PUSH    AX      ; 1
0391 E80000         E     1088               CALL    ENABLE_LINES
                          1089                       ; STATEMENT # 566
                          1090     ;         POP     BP
0394 C3                   1091               RET
0395                      1092     @81:
0395 B004                 1093               MOV     AL,4H
0397 EB4390               1094               JMP     @42
                          1095                       ; STATEMENT # 568
039A                      1096     @83:
039A B0E0                 1097               MOV     AL,0E0H
039C EB2090               1098               JMP     @37
039F                      1099     @84:
                          1100                       ; STATEMENT # 570
039F C606070000     E     1101               MOV     H19_MODE.CURSOR_ON,0H
                          1102                       ; STATEMENT # 571
03A4 E80000         E     1103               CALL    SCP
                          1104                       ; STATEMENT # 573
                          1105     ;         POP     BP
03A7 C3                   1106               RET
03A8                      1107     @85:
03A8 E8C0FE               1108               CALL    P_EKPS
                          1109                       ; STATEMENT # 574
                          1110     ;         POP     BP
03AB C3                   1111               RET
```

```
LOC  OBJ                  LINE     SOURCE

                          1167             PUBLIC  P_COLOR
                          1168     ;       PUSH    BP
                          1169     ;       MOV     BP,SP
                          1170                             ; STATEMENT # 592
03E1 A00600           E   1171             MOV     AL,ESCP.OPERAND
03E4 B130                 1172             MOV     CL,30H
03E6 3AC8                 1173             CMP     CL,AL
03E8 7720                 1174             JA      @96
03EA B237                 1175             MOV     DL,37H
03EC 3AC2                 1176             CMP     AL,DL
03EE 771A                 1177             JA      @96
03F0 A00700           E   1178             MOV     AL,BYTE PTR ESCP+7H
03F3 3AC8                 1179              CMP     CL,AL
03F5 7713                 1180             JA      @96
03F7 3AC2                 1181             CMP     AL,DL
03F9 770F                 1182             JA      @96
                          1183                             ; STATEMENT # 593
03FB A00600           E   1184             MOV     AL,ESCP.OPERAND
03FE 2AC1                 1185             SUB     AL,CL
0400 50                   1186             PUSH    AX      ; 1
0401 A00700           E   1187             MOV     AL,BYTE PTR ESCP+7H
0404 2AC1                 1188             SUB     AL,CL
0406 50                   1189             PUSH    AX      ; 2
0407 E80000           E   1190             CALL    SCA
040A                      1191     @96:
                          1192                             ; STATEMENT # 595
                          1193     ;       POP     BP
040A C3                   1194             RET
                          1195     P_COLOR         ENDP
                          1196                             ; STATEMENT # 596
040B                      1197     P_IDENT         PROC NEAR
                          1198             PUBLIC  P_IDENT
                          1199     ;       PUSH    BP
                          1200     ;       MOV     BP,SP
                          1201                             ; STATEMENT # 598
040B 803E060030       E   1202             CMP     ESCP.OPERAND,30H
0410 7521                 1203             JNZ     @97
                          1204                             ; STATEMENT # 600
0412 E82F00               1205             CALL    TEC
                          1206                             ; STATEMENT # 601
0415 B069                 1207             MOV     AL,69H
0417 50                   1208             PUSH    AX      ; 1
0418 E80000           E   1209             CALL    SXMTC
                          1210                             ; STATEMENT # 602
041B B045                 1211             MOV     AL,45H
041D 50                   1212             PUSH    AX      ; 1
041E E80000           E   1213             CALL    SXMTC
                          1214                             ; STATEMENT # 603
0421 A00300           E   1215             MOV     AL,H19_PAR.POVRAM
0424 0430                 1216             ADD     AL,30H
0426 50                   1217             PUSH    AX      ; 1
0427 E80000           E   1218             CALL    SXMTC
                          1219                             ; STATEMENT # 604
042A A00800           E   1220             MOV     AL,H19_PAR.VRAM_SIZE
042D 0441                 1221             ADD     AL,41H
```

8086/8087/8088 MACRO ASSEMBLER H19_CRT_A86

LOC OBJ LINE SOURCE